Copyright Statement

The digital copy of this thesis is protected by the Copyright Act 1994 (New Zealand).

This thesis may be consulted by you, provided you comply with the provisions of the Act and the following conditions of use:

- Any use you make of these documents or images must be for research or private study purposes only, and you may not make them available to any other person.
- Authors control the copyright of their thesis. You will recognise the author's right to be identified as the author of this thesis, and due acknowledgement will be made to the author where appropriate.
- You will obtain the author's permission before publishing any material from their thesis.

To request permissions please use the Feedback form on our webpage. http://researchspace.auckland.ac.nz/feedback

General copyright and disclaimer

In addition to the above conditions, authors give their consent for the digital copy of their work to be used subject to the conditions specified on the Library Thesis Consent Form and Deposit Licence.
Bioinformatics tools for cell modeling and metabolomics using *Lactococcus lactis* as a model organism

Raphael Bastos Mareschi Aggio

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Biological Sciences. The University of Auckland, New Zealand, 2012
Abstract

This decade’s advances in molecular biology coupled to the need for understanding the cell’s metabolism as a system, established the field of Systems biology. Genome-scale metabolic models joined to high-throughput measurements, such as metabolomics, are largely applied to reconstruct the full picture of the cell’s metabolism and its interaction with the environment. Metabolomics is one of the newest omics technologies, yet it is already considered essential to systems biology studies. Together, systems biology and metabolomics are considered revolutionary fields able to address fundamental questions that enhance our understanding of biological systems. Still, the reconstruction of genome-scale metabolic networks, the processing and interpretation of metabolomics data, and its use for feeding systems biology studies remain limiting factors for these fields achieving their full potential. In this thesis, I present algorithms that I developed to address these limiting factors. To assist the reconstruction of genome-scale metabolic networks, I present a set of algorithms able to merge existing networks and find potential metabolic gaps. To improve the data processing, I present MetaBox. MetaBox deconvolutes chromatogram peaks and identifies metabolites analyzed by gas chromatography–mass spectrometry with at least 20% higher accuracy reported by the most commonly used software for this purpose, namely Automated Mass Spectral Deconvolution and Identification System (AMDIS). In addition, MetaBox automates the normalization steps in, and basic statistical analyses of, metabolomics data. To facilitate the interpretation of metabolomics data, I present Pathway Activity Profiling (PAPi), which generates hypotheses by predicting the activity of metabolic pathways based solely on metabolomics data sets. Finally, I present Metabolite to Network (M2N), which facilitates the usage of metabolomics data for feeding metabolic models by predicting enzymes and reactions related to the organism under study. The usage and the power of these new algorithms are demonstrated using the lactic acid bacteria Lactococcus lactis as a model biological system. Among the significant conclusions related to L. lactis’ metabolism are its potential to undergo respiration by using cyanocobalamin as a heme precursor and a newly reconstructed metabolic network containing previously unseen reactions involving the degradation and biosynthesis of fatty acids in this bacterium. The findings from the metabolism of L. lactis, the algorithms and the computational tools described in this thesis represent novel contributions to the fields of metabolomics and systems biology.
Acknowledgements

Many people contributed to the construction of this thesis.

Thank you Silas for giving me so many opportunities during these last 4 years. I will always be grateful for that.

Thank you Kathy! Thank you for co-supervising me and for supervising me in good moments and also in some of the hardest times of my life. It was an honor to work and laugh with you during these last 4 years. I learned a lot from you and I am sure there is a lot more to come. Thank you!

My friends Luis, Valdeco, Pedrones, Augusto, Thomas (baitolinha) + Karima, Rodrigueira, Luizera, Diegueira, Xavier, Rafa, Morgan, Farhana, Miguelito, Louis, Kathy (again), Simon, Catalina, Gonzo, Francesca, Sergey and many others that shared waves, coffees, food, thoughts, and experiences with me during this long 4 years. Thank you!

Obrigado a toda minha família! Sem o suporte de vocês, mesmo que do outro lado do mundo, eu jamais teria chegado ao fim deste processo. Esta tese jamais haveria saído do mundo das possibilidades! Dedico esta tese a todos vocês com muito amor.

Un agradecimiento especial para mis papas colombianos. Ustedes, Blanquita y Pepo, fueron uno de los más grandes regalos que la vida podría haberme presentado. Ustedes cuidaron de mí como cuidan a sus hijos. Me enseñaron y me consolaron en los momentos más difíciles. Esta tesis es también suya con todo el peso que las palabras tienen. Los amo mucho.

Lastly, thank you Paulina. You are by far my most important achievement in this PhD process. Having met you at our lab was the biggest prize The University of Auckland could have given me. Thank you for sharing your life with me, for being with me at all the good and bad moments, for supervising me, for teaching me how to grow, for taking care of me and for letting me take care of you. This thesis is our thesis! TE AMO.
# Table of Contents

Abstract .................................................................................................................................................. ii
Acknowledgements ................................................................................................................................. iii
List of Figures ........................................................................................................................................ viii
List of Tables .......................................................................................................................................... x

Chapter 1. Introduction ......................................................................................................................... 11
  1.1. Systems biology and genome-scale metabolic models ............................................................... 13
    1.1.1 Genome-scale metabolic networks – bottom-up approach ................................................ 14
    1.1.2 Metabolic gaps ....................................................................................................................... 15
    1.1.3 Constraint-based genome-scale models and Flux Balance Analysis (FBA) ...................... 16
  1.2. Metabolomics .............................................................................................................................. 18
    1.2.1 Sample preparation ............................................................................................................... 20
    1.2.2 Metabolite analysis ............................................................................................................... 21
    1.2.3 Data processing and analysis ............................................................................................... 24
  1.3. Why Lactococcus lactis? ............................................................................................................. 26
    1.3.1 L. lactis’ metabolism ............................................................................................................ 27
  1.4. Thesis aims ..................................................................................................................................... 28
    1.4.1 Reconstructing and feeding metabolic networks ................................................................. 28
    1.4.2 Process and analysis of metabolomics data ......................................................................... 28
  1.5. Thesis outline ............................................................................................................................... 28

Chapter 2. Merging, expanding and finding gaps in genome-scale metabolic networks of
Lactococcus lactis ....................................................................................................................................... 30
  2.1. Introduction .................................................................................................................................. 31
  2.2. Reconstruction method ............................................................................................................... 33
  2.3. Reconstructing the metabolic network of Lactococcus lactis .................................................. 36
  2.4. Results and discussion ............................................................................................................... 47
  2.5. 2.5 Conclusion ........................................................................................................................... 52

Chapter 3. MetaBox - a new algorithm for deconvolution and identification of
metabolites analyzed by GC-MS. ........................................................................................................ 53
  3.1. Introduction .................................................................................................................................. 54
Chapter 6. Metabolite to Network (M2N): a new algorithm for facilitating the use of metabolomics data for feeding metabolic networks. .......................................................... 125

6.1. Introduction ................................................................................................. 126
6.2. M2N – The algorithm ................................................................................ 127
  6.2.1 Input data ............................................................................................... 127
  6.2.2 Description ............................................................................................. 128
  6.2.3 M2N algorithm’s output .......................................................................... 131
6.3. M2N applied to *Lactococcus lactis* IL1403 ............................................. 132
  6.3.1 M2N results for *L. lactis* .................................................................... 135
6.4. M2N Toolbox ............................................................................................. 149
6.5. Conclusion .................................................................................................. 152

Chapter 7. Summarizing discussion and future directions ................................. 153

7.1. Summarizing discussion ............................................................................ 154
7.2. Future directions ....................................................................................... 157

References ........................................................................................................ 158

Appendices ........................................................................................................ 167
  Appendix 2a. ................................................................................................. 167
  Appendix 2b. ................................................................................................. 167
  Appendix 2c. ................................................................................................. 168
  Appendix 2d. ................................................................................................. 169
  Appendix 2e. ................................................................................................. 169
  Appendix 2f. ................................................................................................. 171
  Appendix 2g. ................................................................................................. 172
  Appendix 2h. ................................................................................................. 192
  Appendix 2i. ................................................................................................. 193
  Appendix 2j. ................................................................................................. 194
  Appendix 3a. ................................................................................................. 224
  Appendix 3b. ................................................................................................. 230
  Appendix 3c. ................................................................................................. 232
  Appendix 3d. ................................................................................................. 240
  Appendix 3e. ................................................................................................. 242
  Appendix 3f. ................................................................................................. 243
  Appendix 3g. ................................................................................................. 245
  Appendix 3h. ................................................................................................. 246
  Appendix 3i. ................................................................................................. 247
List of Figures

Figure 1.1. Mathematical representation of a metabolic network.................................................. 16
Figure 1.2. Schematic of organization of omics technologies ...................................................... 18
Figure 1.3. Mass spectrum............................................................................................................ 21
Figure 1.4. The capillary column and the gas chromatography–mass spectrometry (GC-MS) chromatogram........................................................................................................... 22
Figure 2.1. Method for reconstructing genome-scale metabolic networks ..................................... 33
Figure 2.2. Pipeline for reconstructing the genome-scale metabolic network of Lactococcus lactis... 36
Figure 2.3. R function track.metab............................................................................................. 45
Figure 2.4. Integrated metabolic network of Lactococcus lactis .................................................. 49
Figure 2.5. Hubs of the integrated metabolic network................................................................... 50
Figure 3.1. MetaBox - Deconvolution and identification algorithm............................................... 58
Figure 3.2. Correlation between mass fragments.......................................................................... 61
Figure 3.3. Experimental design for MetaBox validation............................................................... 64
Figure 3.4. False negatives.......................................................................................................... 66
Figure 3.5. False positives ........................................................................................................... 67
Figure 3.6. Quantification of metabolites in Mix 1........................................................................ 71
Figure 3.7. Quantification of metabolites in Mix 2........................................................................ 71
Figure 3.8. MetaBox....................................................................................................................... 77
Figure 4.1. Description of PAPi algorithm.................................................................................... 88
Figure 4.2. PAPi’s graph.............................................................................................................. 94
Figure 5.1. Data analysis.............................................................................................................. 104
Figure 5.2. Growth curve, growth rate and pH of Lactococcus lactis IL1403 growing aerobically...... 105
Figure 5.3. Growth curve, growth rate and pH of Lactococcus lactis IL1403 growing anaerobically... 105
Figure 5.4. Metabolites showing different relative abundances for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% galactose ...................... 109
Figure 5.5. Metabolic pathways showing different Activity Scores (AS) for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% galactose ............. 110
Figure 5.6. Metabolites showing different relative abundances for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% glucose........................................ 112
Figure 5.7. Metabolic pathways showing different Activity Scores (AS) for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% glucose ................... 113
Figure 5.8. Metabolite profile of Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% lactose + 1% cellobiose......................................................... 115
Figure 5.9. Metabolic pathways showing different Activity Scores (AS) for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% lactose plus 1% cellobiose ......................... 116
Figure 5.10. Chemical structures of heme, protoporphyrin IX and cyanocobalamin ......................... 118
Figure 6.1. Illustrated description of M2N algorithm ................................................................. 129
Figure 6.2. Enzyme commission numbers (ECs) predicted by M2N and related to the metabolic network of Lactococcus lactis .......................................................... 141
Figure 6.3. Metabolic map generated by M2N ........................................................................... 142
Figure 6.4. Metabolic network produced by M2N ....................................................................... 143
Figure 6.5. Flux balance analysis applied for anaerobic growth when constraining alanine availability and maximizing for biomass formation ............................................................. 147
List of Tables

Table 2.1 – Existent genome-scale metabolic networks of *L. lactis* ................................................................. 37
Table 2.2 – Annotated Genome Sequence of *Lactococcus lactis* ............................................................................. 38
Table 2.3 – *Lactococcus lactis*’ enzyme_list .............................................................................................................. 39
Table 2.4 – KEGG’s reactions ......................................................................................................................................... 40
Table 2.5. Variable fluxes ............................................................................................................................................. 43
Table 2.6 Variable gaps .................................................................................................................................................. 44
Table 2.7. Network features ......................................................................................................................................... 47
Table 3.1. Spectral library ............................................................................................................................................ 56
Table 3.2. MetaBox’ result ............................................................................................................................................ 62
Table 3.3. Two standard samples for MetaBox validation. ........................................................................................... 63
Table 3.4. ANOVA and Tukey’s HSD test results for metabolites in samples originated from Mix 1 ...................... 68
Table 3.5. ANOVA and Tukey’s HSD test results for metabolites in samples originated from Mix 2 ...................... 69
Table 4.1. Input data ...................................................................................................................................................... 87
Table 5.1. Fermentation parameters ............................................................................................................................ 106
Table 5.2. Identified compounds ..................................................................................................................................... 106
Table 6.1. Input data ........................................................................................................................................................ 127
Table 6.2. M2N input data ............................................................................................................................................. 132
Table 6.3. Shortlist ........................................................................................................................................................ 135
Table 6.5. EC numbers representing potential reactions ........................................................................................... 144
Table 6.6. Reactions added to the genome-scale metabolic network of *Lactococcus lactis* ................................. 145
Chapter 1. Introduction
A series of discoveries starting from the elucidation of the DNA structure, in the middle of the last century, established molecular biology as a field. At the mid to late 1990s, it was already possible to sequence the whole genome of an organism and to generate high-throughput measurements (Price et al., 2004). This enabled us to collect comprehensive data sets on underlying molecules such as RNA and proteins. The large volume of data sets generated by these new technologies led to a rapid development of bioinformatics and raised the need for more formal and mechanistic approaches able to analyze these data in a more systemic or holistic manner (Mendes, 2002).

At the same time that molecular biology rapidly progressed, researchers from a different branch of biology were trying to understand the principles and laws governing biological self-organization. Different from the traditional reductionism that predominated at that time, these researchers aimed to discover general principles rather than just being descriptive. They undertook efforts to simulate the integration of molecular functions by combining mathematical modeling with experimental molecular data in a systemic manner (Kitano, 2002). These efforts grew in scale to dynamic simulations of large metabolic networks (Loomis and Thomas, 1976), kinetic models (Otto et al., 1977) and large-scale models of mitosis in early 1990s (Novak and Tyson, 1995). With the advent of genome-scale sequencing, the first genome-scale metabolic constraint-based models were constructed in order to describe functional states or cellular phenotypes (Edwards and Palsson, 1999). In this branch of biology, the ‘genome’ was treated as the ‘system’.

In the beginning of 2000s, there was a convergence of the molecular biology branch with the systemic branch of biology, which originated Systems biology (Kitano, 2002). Systems biology aims for a holistic understanding of the cell’s metabolism; it studies the behavior and relationships of all components within a biological system, while it is functioning (Chuang et al., 2010). For this, omics technologies, such as transcriptomics, proteomics and, more recently, metabolomics are largely applied. Transcriptomics studies the transcriptome, the complete set of RNA transcripts produced by a genome at a given point on time. Proteomics studies the proteome, the entire protein complement expressed by a genome or the protein complement of a complex tissue. Metabolomics studies the metabolome, the complete set of small molecules (≤ 1500 Da), or metabolites, present in a biological sample.

Metabolites are the end-products of regulatory processes occurring inside the cells, and their levels are essentially the ultimate response of biological systems to genetic or environmental
changes (Fiehn, 2002). Compared to the transcriptome and the proteome, the metabolome is
the closest representation of the cell’s phenotype and, for this reason, metabolomics has been
considered the most important omics technology in systems biology studies (Patti et al.,
2012). Together, systems biology and metabolomics have been revolutionizing the ways in
which science is conducted (Chuang et al., 2010; Patti et al., 2012). The cell machinery of
microorganisms was never so well understood and so well under control as it is today (Thiele
et al., 2012; Lerman et al., 2012). Still, systems biology and metabolomics are relatively new
scientific areas with many limitations that must be addressed in order for them achieve their
full potentials.

In this thesis, I introduce methods, algorithms and computational tools I developed to address
specific limitations in metabolomics data processing and its application to systems biology.
In order to illustrate the use of these methods and the results generated by each, I use the
lactic acid bacteria *Lactococcus lactis* as a model organism. Below (sections 1.1-1.3), I
briefly describe essential concepts that allow a better understanding of the work behind this
thesis and its implications.

### 1.1. Systems biology and genome-scale metabolic models

In the mid to late 1990s, the advent of whole-genome sequencing (Fleischmann et al., 1995;
Adams et al., 2000) and the advances in quantitative high-throughput biological tools (e.g.
genomics, transcriptomics and proteomics) allowed the reconstruction of metabolic networks
and global analyses of biological systems’ properties and dynamics (Price et al., 2004; Durot
et al., 2009). A great part of systems biology studies relies on genome-scale metabolic
models to describe features of biological systems and to generate accurate predictions about
their behavior (Price et al., 2004).

Two main approaches are described for modeling in systems biology: top-down and bottom-
up. The top-down approach makes use of heterogeneous high-throughput data (e.g.
transcriptomics and proteomics) generated under distinct environmental and genetic
conditions. Statistical and computational methods are then used to determine the interactions
between components of the system and generate a metabolic network representing the
system’s metabolism (Bonneau et al., 2007). This approach critically relies on the quality of
the tools employed for the measurement of the high-throughput data and in the certainty of
statistical methods. A bottom-up approach models biological systems based on a metabolic
network reconstructed using the annotated genome sequence, biochemical databases and the scientific literature. Although there is active discussion among scientists about the advantages and limitations of each method (Cakir et al., 2009), the bottom-up approach currently has more computational tools, protocols and data sources available to support its development (Thiele and Palsson, 2010). Consequently, it seems to be the most commonly used approach.

1.1.1 Genome-scale metabolic networks – bottom-up approach

A metabolic network is a list of the molecular mechanisms (reactions) and associated molecular components (enzymes and metabolites) that largely define the metabolic capabilities of a cell (Durot et al., 2009). In practice, a metabolic network is a collection of all the potential chemical reactions and transport processes involved in the cell’s metabolism, where chemical reactions are represented by stoichiometric equations. These chemical reactions may be associated with different compartments of the cell, such as the cytosol, extracellular medium, mitochondria or chloroplasts. Thus, transport reactions are needed in the network to simulate the transport of metabolites between the different cellular compartments (Thiele and Palsson, 2010).

A genome-scale metabolic network reconstructed in a bottom-up manner makes use of genome annotations, biochemical databases and the relevant literature to build networks brick-by-brick (Thiele and Palsson, 2010). Genome annotation is the process of adding biological information to raw genome sequences (Stein, 2001) and is usually performed through homology relationships with proteins whose biological functions have been previously elucidated. Databases such as the Kyoto Encyclopedia of Genes and Genomes (KEGG) (Ogata et al., 1999), MetaCyc (Krieger et al., 2004) and Brenda (Pharkya et al., 2003) are rich in biochemical data such as gene sequences, enzymes reversibility and their stoichiometric equations.

Since the availability of automated genome sequencing and its use for systems biology studies, there has been a considerable growth in the number of software and databases related to the reconstruction and analysis of genome-scale metabolic networks. In 2007, for example, the first version of the SEED platform (DeJongh et al., 2007) was released. SEED was developed to automatically reconstruct metabolic networks based on the bottom-up approach. SEED comprises computational tools and biochemical databases that allows users to quickly
annotate genome sequences, reconstruct, curate and analyze metabolic networks. In 2010, Schellenberger and co-workers presented a new concept: the Biochemical Genetic and Genomic (BIGG) knowledgebase of large scale metabolic reconstructions. BIGG is a database of high quality curated metabolic models and reconstructions, which allows users to browse stoichiometric equations in existing networks, visualize metabolic pathways maps and export metabolic networks to be analyzed by external software.

The growing number of databases and platforms developed for metabolic modeling have resulted in an increase in the number of reconstructed metabolic networks. Fifty-nine metabolic networks had been reconstructed until 2009 ([http://gcrg.ucsd.edu/InSilicoOrganisms/OtherOrganisms](http://gcrg.ucsd.edu/InSilicoOrganisms/OtherOrganisms)) and this number is increasing. They represent 39 organisms, indicating that some organisms are represented by more than one metabolic network. For example, there are two genome-scale metabolic networks representing the lactic acid bacterium *Lactococcus lactis* (Oliveira et al., 2005; Oddone et al., 2009). Although these networks were both reconstructed based on extensive literature curation, they contain different configurations of stoichiometric equations for describing the same biochemical reaction and reactions that are present in one network but not the other. Thus, in order to expand the metabolic network of *L. lactis*, for example, one must combine and update the data contained in the existent networks in order to ensure a metabolic network as complete and precise as possible. However, manually verifying the configuration of about 1000 stoichiometric equations is considerably time consuming and error-prone. Therefore, methods for automating, or at least semi-automating, this process of expanding metabolic networks are urgently needed.

**1.1.2 Metabolic gaps**

The substrates of every reaction must be the product of a different reaction or they must be transported from a different network compartment. Also, the products of every reaction must be the substrate of a different reaction or they must be transported to another compartment. Reactions involving substrates not produced or products not consumed may result in erroneous results when simulating the system. Most automatically reconstructed metabolic networks are incomplete (Kumar et al., 2007). They miss steps of metabolic pathways, called *metabolic gaps*, which ultimately lead to missing functionalities in a biological system. In practice, metabolic gaps are evident in the network as substrates that are not produced and/or products that are not consumed.
Visualizing an entire network containing a large number of reactions (e.g. 1000) in search of metabolic gaps is a considerably arduous, time-consuming and inefficient task. Therefore, the development of algorithms to find gaps and their implementation in computer programs which provide bioinformatics tools to automatically indicate metabolites and reactions potentially involved in metabolic gaps are essential. Gapfind (Kumar et al., 2007) is perhaps the most popular bioinformatics tool for this purpose. However, at the time the work described in this thesis was being conducted, using Gapfind required expensive commercial mathematical solvers and extensive data reformatting. Therefore, alternative methods using more accessible tools and in a high-throughput manner were required.

1.1.3 Constraint-based genome-scale models and Flux Balance Analysis (FBA)

Modeling a biological system is mostly achieved by constraining its metabolic network in order to reduce the number of potential phenotypes or metabolic states (Durot et al., 2009). Different types of constraints can be used to limit cellular function; for example, the stoichiometry of reactions and their reversibility are constraints intrinsic to a metabolic network. Additional constraints such as nutrient availability, pH and the availability of electron acceptors can also be applied (Thiele and Palsson, 2010). Then, based on the genome-scale metabolic network and its constraints, the metabolic state of a cell can be described by metabolite concentrations and reaction rates. Flux Balance Analysis (FBA) is a popular method for doing this (Orth et al., 2010).

FBA is a method for steady-state flux analysis developed specifically to predict quantitative growth phenotypes (Varma and Palsson, 1994). It is able to predict the quantitative relationships between nutrients’ uptake rates, growth rates and by-products’ output rates (Edwards et al., 2002). FBA does this by first converting a metabolic network into a numerical matrix \( S \) of the stoichiometric coefficients of each reaction in the metabolic network (Figure 1.1). The matrix is of size \( m \times n \), where \( m \) rows correspond to metabolites and the \( n \) columns correspond to reactions. Column entries correspond to stoichiometric coefficients of the metabolites participating in a specific reaction, with negative coefficients indicating metabolite consumption and positive coefficients indicating production. Metabolites that are not participating in a reaction are given a stoichiometric coefficient of 0. The stoichiometric coefficients of metabolites enforce balance constraints to the flow of
metabolites through the metabolic network, ensuring that the amount of any compound being produced equals the total amount being consumed (Orth et al., 2010). This feature is based on Lavoisier’s fundamental law of mass conservation, which is at the core of FBA. At steady state, the accumulation or depletion of metabolites in a metabolic network are not expected. Therefore, the rates of production and consumption of each metabolite in the network must be equal (Orth et al., 2010). The concentration of a metabolite per time unit can be represented by the sum of fluxes of all reactions that involve production and consumption of this specific metabolite. In addition, every reaction in the network may receive lower and upper bounds, defining the minimum and maximum possible fluxes, respectively, through each reaction.

Figure 1.1. Mathematical representation of a metabolic network. The metabolic network is converted into a numerical matrix S (A and B), where rows represent metabolites and columns represent reactions. The entries in this matrix correspond to the stoichiometric coefficient of each compound in each reaction. Positive coefficients indicate production of compounds, while negative coefficients indicate consumption. A value of 0 is inserted when the compound is not involved in a reaction. At steady state, the concentration of a metabolite per unit time can be represented by the sum of the fluxes of every reaction involving production and consumption of this specific metabolite (C).

The fluxes through all the reactions in the network are represented by a vector (v), which are converted into a system of mass balance equations, Sv = 0, which are subsequently solved by linear programming methods. Since genome-scale metabolic networks generally contain more reactions than compounds (i.e. n > m), the system of mass balance equations contains more unknown variables than equations. Such systems do not have a unique solution, i.e. more than one possible solution, or phenotype, is possible. The set of all possible solutions forms the solution space. Together, the stoichiometric coefficient of each metabolite and the lower and upper bounds of each reaction constrain the solution space or phenotype produced by the metabolic network (Orth et al., 2010). Additional constraints, such as temperature, pH
1.2. Metabolomics

Metabolomics is a relatively new omics technology that aims to study the impact of environmental and/or genetic perturbations on the metabolome – the complete set of small molecules (≤ 1500 Da), or metabolites, present in a biological sample at a given point in time. Metabolites are intermediates of biochemical reactions, and their concentrations depend mostly on the levels and properties of enzymes. Consequently, metabolites’ levels result from complex interactions involving many regulatory processes occurring within the cells (e.g. regulation of transcription, translation and protein-protein interaction).

In a hypothetical hierarchical organization of omics technologies, metabolomics would be located at the bottom of the list (Figure 1.2): metabolites concentrate the changes along the flow of information from gene expression and represents the most downstream effects of genetic or environmental perturbations. The profile of metabolites - a list of metabolites identified as associated with the cell’s metabolism and their respective levels - within a cell is the result of the interaction of the cell’s genome with its environment so that, ultimately, the level of metabolites represents the phenotype of a biological system (Fiehn, 2002). When combined with metabolic networks, the metabolite profile is considered key information for understanding the regulation of a cell’s metabolism, and this makes metabolomics one of the most relevant omics technologies applied to systems biology studies (Patti et al., 2012).
The identification and measurement of metabolites levels *per se* is not something developed by or with metabolomics. In 1776, Matthew Dobson began identify metabolites present in urine. Since then, identification and measurement of metabolites have been largely applied in medicine and today this is part of the daily routine of hospitals and clinics for diagnostics. The innovation that may be accredited to metabolomics is the approach of simultaneously analyzing several metabolites and comparing their levels under different experimental conditions.

The ultimate goal of metabolomics is to quantify all of the metabolites in a cellular system in a given state and at a given point in time. However, the high number of metabolites and their enormous diversity make such quantification virtually impossible (Goodacre et al., 2004). Sample preparation processes such as *metabolite extraction* – obtaining metabolites present inside the cells – and *chemical derivatization* – turning non-volatile metabolites into volatile ones – are quite specific to certain classes of metabolites (e.g., organic acids). Therefore, the main class of metabolites reported by metabolomics experiments is determined by the laboratory protocols that are applied, biasing the information produced by metabolomics experiments.

Metabolomics experiments are generally performed using two main approaches: *fingerprinting* and *footprinting*. Fingerprinting involves the analysis of the *endometabolome*.

Figure 1.2. Schematic of organization of *omics* technologies. The flow of information goes from genes to transcripts to proteins to metabolites. The dashed lines indicate regulatory interactions between metabolites, proteins, RNA and DNA.
(intracellular metabolites), while footprinting aims to analyze the exometabolome (extracellular metabolites) present in the spent culture medium (Nielsen and Oliver, 2005). In both cases, metabolites do not necessarily need to be identified. The “raw” data generated by the analytical method applied (e.g. GC-MS) can be used to determine potential differences in the cell’s metabolism under different experimental conditions without knowing the identity of metabolites.

A typical metabolomics experiment applied to microbial cells can be divided into three main steps: 1) sample preparation, 2) metabolite analysis and 3) data processing and analysis.

1.2.1 Sample preparation

Ideally, metabolomics analysis of a sample reveals the metabolic state of cells at the exact moment in which a sample is harvested. However, many important metabolites involved in cell metabolism, such as ATP and NADH, can be quickly catalyzed by enzymes or degraded (< 1 mMs⁻¹) when exposed to factors such as high temperature and light. Consequently, the levels of these metabolites change very quickly during sample preparation and so do their final concentrations, resulting in metabolic profiles that may not represent the cell’s true metabolic state (Villas-Boas and Bruheim, 2007).

To acquire more reliable metabolomics results, minimizing continuation of the cell’s metabolism, the sample must be quickly quenched prior to any further sample handling. When analyzing the endometabolome, metabolites must be extracted from the cell, which is achieved by combining the use of an agent, usually an organic solvent, and a method to make cell’s membrane more permeable. Extracellular metabolites are generally obtained by filtration or centrifugation (Smart et al., 2010). Different analytical methods [e.g. gas chromatography–mass spectrometry (GC-MS), Nuclear magnetic resonance (NMR) and Liquid chromatography–mass spectrometry (LC-MS)] can be used individually or in combination to analyze metabolites. When analyzed by GC-MS, metabolomics samples must be subjected to chemical derivatization. There are two main types of chemical derivatization: silylation and alkylation. Briefly, sugars and sugar-derivatives are better derivatized by silylation, while organic acids are better derivatized by alkylation (Smart et al., 2010).
1.2.2 Metabolite analysis

Several analytical platforms can be used to analyze metabolites in biological samples: gas chromatography (GC), gas chromatography coupled with mass spectrometric detection (GC-MS), capillary electrophoresis (CE), liquid chromatography coupled with mass spectrometric detection (LC-MS), direct infusion mass spectrometry (DIMS), Fourier transform-infrared (FT-IR) and nuclear magnetic resonance spectroscopy (NMR). Each of these platforms has advantages and limitations when analyzing specific classes of metabolites or analytes (e.g. sugars and organic acids). Therefore, different analytical platforms may be combined in order to cover a wider range of metabolites present in biological samples (Dunn and Ellis, 2005).

GC-MS is considered the most popular platform in metabolomics (Mendes, 2002), as it is able to separate complex mixtures of metabolites with high efficiency at relatively low costs (Villas-Bôas et al., 2011). Most of the metabolomics tools introduced in this thesis were developed to exclusively analyze data generated by GC-MS. Therefore, a more detailed description about GC-MS operation is provided below.

Gas chromatography–mass spectrometry (GC-MS)

GC-MS is a hyphenated analytical technique that combines gas chromatography (GC) and mass spectrometry (MS) to analyze metabolites present in a mixture of chemicals. The gas chromatography separates metabolites in a mixture (Smedsgaard, 2007), while the mass spectrometer fragments and detects each of the metabolites (Smedsgaard, 2007).

When a mixture of metabolites is injected into the GC-MS, it firstly enters the GC, which is composed of a capillary column (stationary phase) inside a computer-controlled oven (Smedsgaard, 2007). The capillary column is made of chemicals able to selectively attract metabolites in a mixture while the oven vaporizes metabolites injected into the capillary column. Entering the GC, vaporized metabolites are then carried through the capillary column by a mobile phase, which is generally an inert gas such as helium. While travelling through the capillary column, metabolites are separated according to their interaction with the stationary phase, whereby metabolites with weaker interactions travel faster (Smedsgaard, 2007).
Metabolites eluting from the GC enter the ion source. One of the main functions of the ion source is the ionization of metabolites, and the most common ionization techniques are called electron impact ionization (EI) and electrospray ionization (ESI). When using EI, metabolites enter the ion source and are bombarded with a stream of electrons that break them into ion mass fragments (IMFs) (Smedsgaard, 2007). These fragments eventually reach the recorder, a computer that registers or scans the mass to charge ratio \((m/z)\) and the current or intensity of each fragment (Smedsgaard, 2007). Each scan performed by the detector generates one spectrum. The number of scans per second determines the time resolution of the GC-MS and varies according to the GC-MS equipment and the method desired. For example, the GC-MS can be configured to perform scans every 0.1 seconds. A metabolite fragmented by EI is expected to always have the same fragmentation pattern, and the intensities of its IMFs tend to always be in the same relative proportions. (Smedsgaard, 2007). For example, the amino acid alanine generates the IMFs 102 m/z, 59 m/z, 58 m/z, and 70 m/z; the intensities of 59 m/z, 58 m/z and 70 m/z relative to the IMF 102 m/z are nearly always 0.184, 0.146 and 0.105, respectively. The fragmentation pattern of each compound is stored in a mass spectrum, a list of \(m/z\) with their respective intensities (Figure 1.3).

![Figure 1.3. Mass spectrum.](image)

Each metabolite analyzed by gas chromatography–mass spectrometry (GC-MS) is represented by a mass spectrum, which describes the fragmentation pattern of each metabolite and the intensity ratio between mass fragments.

A metabolite analyzed by GC-MS takes a specific time to travel through the gas-chromatography column, called the retention time (RT) (Smedsgaard, 2007). Although environmental conditions (e.g. humidity, the age of the capillary column, etc.) and other...
factors may shift the RT of each compound, a specific metabolite is expected to show approximately the same RT when analyzed by GC-MS. Therefore, each compound can be identified by its fragmentation pattern and its approximate RT.

As described above, when a compound elutes, or is released, from the capillary column, it is bombarded by electrons and broken into fragments that eventually reach the recorder. However, not every molecule of this specific compound elutes from the capillary column at the same time; molecules start to be released at low quantities, increasing until they reach their maximum concentration, and then decreasing again with the last few molecules. Consequently, if one plots a chromatogram (the intensity detected for a fragment, or a group of fragments, plotted in relation to time), each analyte will be ideally represented by a single peak (Smedsgaard, 2007) (Figure 1.4).

![Chromatogram](image)

**Figure 1.4.** The capillary column and the gas chromatography–mass spectrometry (GC-MS) chromatogram. Molecules travel through the capillary column of the gas chromatograph and start eluting at low and increasing concentration until they reach the maximum concentration followed by a trail with the last few molecules. These molecules are then fragmented and finally reach the recorder, which ultimately generates the chromatogram. The peak represented in the chromatogram is a result of the number of molecules from the same metabolite eluting from the capillary column per unit time.

If the different metabolites in a mixture are well separated when travelling through the GC, a single peak of the chromatogram is likely to contain all fragments of a single metabolite. However, if the GC separation is not effective, different metabolites coelute (elute from the capillary column simultaneously) and fragments from these metabolites will reach the
recorder at approximately the same time (Smedsgaard, 2007). The result will usually be a single peak on the chromatogram representing fragments originating from different metabolites, making their identification considerably difficult. In these cases, a process called deconvolution is applied, and fragments of each metabolite are analyzed separately so as to distinguish multiple metabolites represented by the same peak.

1.2.3 Data processing and analysis

Data processing

The analytical platforms used in metabolomics generally produce a computer file containing information in a format that is not meaningful for humans without further processing. For example, GC-MS produces a computer file containing a numerical matrix representing lists of m/z with their respective intensities recorded along the analysis of a specific biological sample. Depending on the GC-MS method used, more than 3000 lists of m/z is generated for a single sample. Each list may represent one or more metabolites and must be individually analyzed in order to identify and compare the level of metabolites or ion mass fragments across biological samples. Consequently, interpreting metabolomics data based solely on the raw data generated by GC-MS is not practical. GC-MS data must be translated into a format that is more accessible to analyze and interpret.

Two main approaches can be used to translate GC-MS data into an accessible format: non-target and target. The non-target approach translates GC-MS data into a list of mass fragments with their respective intensities. There is no identification of metabolites. This approach is useful for classifying samples harvested under different experimental conditions or for finding mass fragments which are differentially abundant between the conditions tested (Smart et al., 2010). Software such as GeneSpring (www.home.agilent.com) and R-software (R Development Core Team, 2012), coupled to packages such as XCMS (Smith et al., 2006), are able to apply this type of data processing. The target approach involves the deconvolution of GC-MS chromatograms, identification and quantification of metabolites. As a result, the spectra recorded by the GC-MS system are translated into a list of metabolites and abundances or intensities. The mass spectra contained in the GC-MS file are compared against a mass spectral library of known metabolites. This library is built using chemical standards previously derivatized and analyzed by GC-MS. Positive matches between mass
spectra contained in the GC-MS file and mass spectra from the spectral library result in positive identification of metabolites. In a chromatogram, each chromatographic peak ideally represents a single metabolite and its quantification is generally performed using the area below its respective peak (i.e. area under the curve) or by recording the intensity of the most abundant IMF and its associated RT.

Manually performing target analysis is extremely laborious and time-consuming. Therefore, software such as Automated Mass Spectral Deconvolution and Identification System (AMDIS) (Stein, 1999) has been used to process GC-MS data. Although AMDIS was originally developed for identifying chemical weapons in complex chemical mixtures, it has been considered the most commonly used freeware in metabolomics (Hamzehzarghani et al., 2008; Carneiro et al., 2011). However, when applied to metabolomics studies, AMDIS has three major limitations. First, it selects different mass fragments for quantifying metabolites across samples (Smart et al., 2010), preventing the comparison of metabolite levels between different experimental conditions. Second, it produces a considerably high number of false positives (Behrends et al., 2011), potentially misleading the biological interpretation of metabolomics studies. Finally, third, AMDIS reports result in a data layout that does not make it amenable to further processing and analysis (Smart et al., 2010). In consequence, data processed by AMDIS often require extensive manual correction, filtering and reorganization.

Metabolomics data sets generated by GC-MS need normalization, which vary depending on the objective of the study. Data normalizations by internal standard, biomass and medium content are the most commonly applied in microbial metabolomics (Smart et al., 2010). Data normalization in metabolomics may require a considerable amount of time if performed manually.

In summary, there is an urgent need for computational tools able to reliably deconvolute GC-MS chromatograms, identify and quantify metabolites in a high-throughput manner, perform data normalization, and report results in a layout that allows further data analysis.

Data analysis and interpretation

After data processing, typical metabolomics dataset consists of a list of identified metabolites, or mass fragments, and their respective levels across the different samples analyzed.
According to the design and the aim of the experiment, finding metabolites, or mass fragments, showing significantly different levels across experimental conditions may be sufficient to identify the biological meaning of the results. Unsupervised statistical methods such hierarchical cluster analysis (HCA) and principal component analysis (PCA) may be used to explore patterns in the data, while analysis of variance (ANOVA) and pair-wise *t*-tests may be used to draw inferences on the estimated differences in metabolite levels between different experimental conditions. Supervised methods, such as Fisher discriminant analysis, are generally applied for classifying samples based on prior knowledge (Sue et al., 2011).

Most metabolomics experiments applied to systems biology aim to reconstruct a broader picture of the cell’s metabolism in order to understand how it reacts in response to the genetic and/or environmental variables being tested. In these cases, association between the level of metabolites and the activity of metabolic pathways is generally desired. However, the level of each metabolite results from many regulatory processes occurring inside the cells (e.g. transcription, translation and protein-protein interaction). These processes are unique to the physiological state of the cell at the exact moment the samples are harvested. In addition, microbial cells have more metabolites than enzymes so that the same metabolite can participate in many different pathways. Together, these factors make the correlation between metabolite levels and metabolic pathway activity one of the most difficult ‘omics’ data to interpret (Villas-Bôas et al., 2005). Indeed, few tools are available for assisting biological interpretation of metabolomics data (e.g. MetPA, Pathway Hunter Tool and Ingenuity Pathway Analysis); however, most of them require extensive data pre-processing and demand substantial knowledge of cellular metabolism. Together, the factors described above make biological interpretation one of the major bottlenecks in metabolome analysis (Çakir et al., 2006).

1.3. Why *Lactococcus lactis*?

*Lactococcus lactis* are lactic acid bacteria largely used in the dairy industry. They are one of the main bacteria species used as starter culture for fermented food products such as cheese and fermented milk (Kowalczyk et al., 2008). The dairy industry is responsible for more than 20% (US$26.6 billion) of the merchandise exports of New Zealand (New Zealand
Government. Global New Zealand International Trade, 2012), which represents a significant fraction of the country’s profits. Therefore, new discoveries about L. lactis’ metabolism may provide insights into the potential improvement of dairy products, which are essential to maintaining New Zealand’s competitiveness in international markets.

L. lactis is also considered one of the most studied lactic acid bacteria (Oliveira et al., 2005). Much is known about its physiology, regulatory mechanisms and metabolic pathways; its genome is completely sequenced and annotated. Furthermore, two genome-scale metabolic networks have been reconstructed for this organism. Altogether, these factors make L. lactis a very attractive model organism to illustrate the bioinformatics tools I introduce in this thesis.

1.3.1 L. lactis’ metabolism

L. lactis are gram-positive bacteria able to quickly convert sugar into lactic acid. This species is able to convert over 90% of the carbon source available in the medium mainly into lactic acid and only about 5% into biomass (Novak and Loubiere, 2000). L. lactis are considered “nutritionally fastidious” bacteria (Oliveira et al., 2005) and have a very limited biosynthetic capacity. Thus, anabolic precursors are primarily imported from the medium and a small part is synthesized de novo from carbon sources. L. lactis requires exogenous supplies of at least six amino acids and three B-vitamins. For optimum growth, they require carbon sources, vitamins, amino acids, phosphate, potassium and magnesium (van Niel and Hahn-Hagerdal, 1999).

L. lactis are classified as facultative anaerobic bacteria. Some strains are able to grow in the presence of oxygen while others are strongly inhibited (Oliveira et al., 2005). They are able to undergo respiration provided that heme is added to the culture (Duwat et al., 2001). In the absence of oxygen, L. lactis can use different carbon sources to produce mainly lactic acid (Duwat el al., 2001) while oxygenation of cultures shifts the sugar fermentation toward mix acid fermentation. This results in the production of by-products such as acetic acid, formic acid, CO2, ethanol, acetoin and lactic acid (Smart and Thomas, 1987).
1.4. Thesis aims

1.4.1 Reconstructing and feeding metabolic networks

The first aim of this project is to address current limitations in reconstructing and expanding metabolic networks by: 1) developing a method able to merge existing metabolic networks and 2) developing a method for using metabolite profiles for feeding metabolic networks.

1.4.2 Process and analysis of metabolomics data

The second and most significant aim of this project is to address current limitations in processing and analyzing GC-MS-generated data by: 1) developing an algorithm able to reliably deconvolute GC-MS chromatograms, thereby identifying and quantifying metabolites, 2) generate algorithms able to process metabolomics data in a flexible and high-throughput manner and 3) developing a method able to assist in the biological interpretation of metabolomics data by predicting the activity of metabolic pathways using metabolite profiles.

1.5. Thesis outline

Following this introduction (chapter 2) I describe methods and algorithms for semi-automating the merging of genome-scale metabolic networks and finding potential metabolic gaps. An end result of this chapter is a combined metabolic network of *L. lactis* to be further extended using the metabolomics data produced in the following chapters. In chapter 3, I introduce a new algorithm for improving the deconvolution and identification of metabolites analyzed by GC-MS, and an R package for high-throughput metabolomics data processing. In the next chapter, I present a new algorithm along with an R package that generate hypotheses based solely on metabolomics data. In chapter 5, I use the algorithms described in chapters 3 and 4 in order to produce metabolite profiles of *L. lactis* growing on different carbon sources and oxygen availability. In chapter 6, I introduce a new algorithm for assisting the use of metabolomics data for feeding and expanding metabolic networks. In order to illustrate its usage, the metabolite profiles generated in chapter 5 are used as input data to expand *L.
lactis’ metabolic network produced in chapter 2. Lastly, in chapter 7 I provide a summarizing discussion of the findings, implications and future directions of the work presented herein.
Chapter 2. Merging, expanding and finding gaps in genome-scale metabolic networks of *Lactococcus lactis*
2.1. Introduction

The last decade has seen genome sequencing technologies improve at an astonishing rate (Mardis, 2011). The cost of sequencing a whole genome is now 100,000 times cheaper than it was ten years ago (National Human Genome Research Institute, http://www.genome.gov/sequencingcosts/). A similar trend has also been observed in genome annotation, the process of adding biological information to genome sequences (Stein, 2001). Furthermore, investments and collaborations between different research groups [e.g. European Bioinformatics Institute (EBI), Bioinformatics Centre of Kyoto University and Human Genome Centre of the University of Tokyo] have resulted in the development of biochemical databases, such as the Kyoto Encyclopaedia of Genes and Genomes (KEGG) (Ogata et al., 1999), Brenda (Pharkya et al., 2003) and MetaCyc (Krieger et al., 2004), which contain enormous quantities of information about genes, proteins, metabolites, metabolic pathways and their regulation. Never before has biochemical information been more accessible than it is today.

Together, the annotated genome sequence, biochemical databases and the literature, are the main sources of information to reconstruct genome-scale metabolic networks (Thiele and Palsson, 2010). By homology relationship with proteins of known biological function, the annotated genome sequence is used to predict enzymes potentially playing part in the metabolism of a specific organism. Then, the literature and biochemical databases are used to identify the stoichiometric equations and physiological parameters (e.g. pH and temperature) associated with each predicted enzyme. It is necessary to determine which substrates and products are involved in each reaction, their stoichiometric coefficients, the reaction reversibility, and the cell compartment where each reaction takes place (e.g. cytoplasm, mitochondria, chloroplasts, etc.) (Thiele and Palsson, 2010).

Currently, 59 genome-scale metabolic networks have been reconstructed (http://gcrg.ucsd.edu/InSilicoOrganisms/OtherOrganisms). Analysis of these networks, mostly using constraint-based techniques, has become essential to systematically study the metabolism of microorganisms (Price et al., 2004; Thiele and Palsson, 2010). For some bacteria, such L. lactis, more than one genome-scale network has been reconstructed (Oliveira et al., 2005; Oddone et al., 2009). Due to the complexity of the reconstruction process and the several distinct sources of literature used, each of these networks has its own particularities and, thus, must be considered individually. In order to expand these metabolic
networks towards different metabolic pathways, one must ideally compare and merge their reactions to fully explore the information available for a specific organism. However, manually merging metabolic networks containing about 1000 reactions is time consuming and may result in additional human errors. Therefore, a method able to automate or semi-automate merging of metabolic networks is needed.

Once the metabolic network is reconstructed, the cell’s metabolism can be further analyzed using constraints-based techniques such as Flux Balance Analysis (FBA) (Varma and Palsson, 1994), a technique used to simulate the metabolic state of a cell while growing under steady-state conditions. FBA predicts the flux distribution through the metabolism of an organism when the cell populations are growing under different environmental conditions (Oberhardt et al., 2009). FBA analysis is commonly coined as ‘simulation’ because it uses a model to predict the metabolic state of a cell. To enable this, FBA requires that the metabolic networks to which it is applied contain no gaps, i.e. every substrate of an intracellular reaction must be the product of at least one other intracellular reaction, or it must be transported from the extracellular into the intracellular “compartment”. Similarly, all products of intracellular reactions must be consumed by other intracellular reactions and/or transported to the extracellular compartment (Thiele and Palsson, 2010).

Gap finding is an arduous and time-consuming task when analyzing a metabolic network containing a large number of reactions, e.g. 1000 or more. It is therefore impossible to visualize the whole network in order to identify metabolic gaps. Consequently, each individual reaction must be inspected to verify whether its substrates and products are produced and/or consumed by other reactions. Performed manually, these verifications can be considerably inefficient. Therefore, bioinformatics tools for gap-finding automation are required.

In this chapter, I present a new method that aims to: 1) merge, 2) expand and 3) find gaps in metabolic networks. This method makes use of the annotated genome sequence, KEGG database and FBA. In order to demonstrate its usage, I reconstructed a new version of the genome-scale metabolic network of L. lactis. In addition, this method was semi-automated by implementing each of its steps in functions developed in R (R Development Core Team, 2012) and MATLAB (MATLAB 7.10.0; R2010a).
2.2. Reconstruction method

In this section I describe a new method of reconstructing genome-scale metabolic networks using existing metabolic networks, the Annotated Genome Sequence (AGS) and the KEGG database.

The first step in this method consists of comparing and integrating two existent metabolic networks (Figure 2.1 – Step 1). I compare the configuration of every single stoichiometric reaction described in both networks. This enables identification of reactions related to the same enzyme but showing different stoichiometry in each network, as well as reactions only described in one network. Reactions showing the same configuration in both networks and those described in only one of the two networks are immediately selected for the final integrated network. Reactions showing stoichiometric discrepancies between networks are first corrected, using supporting literature and web databases, and then added to the final integrated network. As a result, I obtain an integrated genome-scale metabolic network.

The AGS of an organism contains all the potential genes identified in the genome sequence of a specific organism. For example, the AGS of *L. lactis* contains 2266 genes annotated. For each gene it describes: the identification code for its locus (first column), its predicted name (second column), its putative biological function (third column) and, finally, its position at the 5’ and 3’ ends of the DNA sequence (last two columns). For some genes, the putative biological function contains their respective Enzyme Commission (EC) number, a numerical classification used to define the type of chemical reaction catalyzed by an enzyme (Bairoch, 2000). As the EC number and the gene locus ID are used by databases, such as KEGG database, to store information about enzymatic reactions, they are very useful for collecting details about stoichiometric equations. Thus, the second step of this method adds to the integrated network the locus ID of each gene and its potential biological functions present in the AGS, including EC numbers.

In the third step, I use the locus ID and the EC number of each gene to obtain stoichiometric equations from the KEGG database. This way, I obtain from the KEGG database the stoichiometry of every reaction related to the genes identified for the organism under study and their respective EC numbers. Reactions obtained from KEGG are useful for two reasons: 1) they can be used as a third reference for defining the correct stoichiometry of reactions
differently described in existing metabolic networks, and 2) they represent potential reactions to be added to the metabolic network; collecting reactions from KEGG database can be particularly useful when dealing with not-so-well studied organisms, where there is a potential lack of reactions related to it in biochemical databases. I then use the literature to confirm or refute the presence of these reactions in the integrated metabolic network. Thus, the third step generates the first draft of the genome-scale metabolic network.

**Figure 2.1. Method for reconstructing genome-scale metabolic networks.** I used existing genome-scale metabolic networks, the AGS and KEGG to combine, and then expand, two existing metabolic networks of *Lactococcus lactis*. In the first step (Step 1), for each reaction, I compared the configurations of the stoichiometric equations between the existing metabolic networks. Consistent configurations were immediately added to the new network; inconsistencies were corrected using the literature before being added. As a result, the first step generated an integrated metabolic network. The description and the locus ID of each gene present in the Annotated Genome Sequence (AGS) of *L. lactis* were inserted into the integrated network (Step 2). The gene description contains putative Enzyme Commission (EC) numbers associated with each gene. The EC number and the locus ID are commonly used as identification for enzymes in biochemical databases and, thus, were used to expand the metabolic network by obtaining from KEGG database potential reactions playing part in the metabolism of *L. lactis* (Step 3). These reactions were inspected using the literature and reactions showing no indication of involvement in the metabolism of *L. lactis* were removed. I used Flux Balance Analysis and the literature to find and fill metabolic gaps in the integrated network (Step 4).
The metabolic network reconstructed as described above may contain metabolic gaps, i.e. reactions containing substrates and/or products not produced or not consumed by other reactions. Therefore, in the fourth and final step of this method I locate and fill these metabolic gaps. To this end, I developed a new algorithm that utilizes FBA for reporting reactions and metabolite potentially involved in metabolic gaps. FBA is a widely used approach for calculating the flow of metabolites through a metabolic network, thereby allowing the prediction of growth rate of an organism and the rate of production of a metabolite (Orth et al., 2010). FBA can be used to optimize metabolic networks to maximize or minimize the product of specific reactions. For example, when optimized for maximum biomass production, the fluxes through the network are distributed in a manner that result in the highest flux through the reaction representing biomass formation. The same approach can be applied to any reaction in the network by setting this specific reaction as objective function.

As FBA is based on mass balance equations, if the reaction chosen for the objective function is involved in metabolic gaps, which means that at least one of its substrates and/or products is not being produced or consumed by other reactions, the FBA simulation will return a value of 0. In this case, if the metabolite that is not being produced or consumed is removed from the reaction chosen for the objective function, FBA will then return a value that differs from 0, which indicates that the objective equation is no longer involved in metabolic gaps. My algorithm for gap finding then uses this characteristic of FBA to identify reactions and compounds potentially involved in metabolic gaps. Firstly, I release any flux constraint in the network, which is achieved by allowing unconstrained flux through every reaction. Then, I apply FBA for maximizing one single reaction at a time, until I have tested every reaction of the metabolic network. Reactions returning the value of 0 when maximized are considered as potentially involved in metabolic gaps. However, any intermediate of these reactions could be the responsible for this metabolic gap. Thus, I reapply FBA for maximizing reactions that returned the value 0, however, removing one intermediate at a time. For example, if FBA returned the value 0 when maximizing the reaction \(A + B \rightarrow C + D\), I modify this reaction by removing the intermediate A and reapply FBA to the resultant reaction \(B \rightarrow C + D\). If FBA again returned the value of 0, the intermediate A is then considered as not involved in gaps. However, if FBA resulted in a value that differed from 0, it indicated that the compound A is not being produced by another reaction in the metabolic network. Doing so, I identify every reaction involved in metabolic gaps, highlighting the metabolites responsible for these.
Once the reactions and metabolites involved in metabolic gaps are identified, I then use the literature and the metabolic databases KEGG, MetaCyc and Brenda to understand the metabolism of each compound and find additional reactions able to fill each metabolic gap. Step 4 (Figure 2.1 – Step 4) is repeated until no metabolic gaps are reported.

2.3. Reconstructing the metabolic network of Lactococcus lactis

The reconstruction algorithm described in Section 2.2 is considerably time-consuming when performed manually. Therefore, I automated this algorithm using two computing environments, namely R and MATLAB. While, ideally I would have implemented the entire algorithm in a single computing environment, it was not possible for two main reasons: there is no R package yet developed for FBA analysis; and MATLAB functions for handling files from Microsoft Excel (XLS) are not functioning in Macintosh Operating System (MacOS), which is the operating system I use. Therefore, I used MATLAB for FBA and R for handling operations on XLS files.

Below, I describe in detail each step of the reconstruction of L. lactis’ genome-scale metabolic network (Figure 2.3). I used Courier New as type font when relating to a computational function, their arguments or required files.

First, I downloaded the networks of Oddone et al., (2009) and Oliveira et al., (2005), available as supplementary material of their respective scientific articles, and reorganized them as partially shown in Table 2.1. Each row of these networks corresponds to one biochemical reaction, with each column describing its parameters. The first parameter is the reaction identifier (RXNID), an alphanumeric character string comprising the gene name plus the number of reactions related to this gene, such as “accA_1” for Acetyl-CoA carboxylase. RXNID is used to uniquely identify each reaction. The EQUATION and SUBSYSTEM columns contain the stoichiometric equation (e.g. D-glucose-6-phosphate <=> D-fructose-6-phosphate) and metabolic pathway where each reaction occurs (e.g. Glycolysis). Once these networks were reorganized, I stored them in R data frames named net1 and net2, respectively. Technically, a data frame in R is a type of object. In practice, a data frame is a type of table that can contain numeric and character values distributed in rows and columns.
Figure 2.2. Pipeline for reconstructing the genome-scale metabolic network of *Lactococcus lactis*. The pipeline for reconstructing the metabolic network of *Lactococcus lactis* involved 4 steps. Firstly, in a data frame, an R object that is a table containing numeric and character values distributed in rows and columns, called net12 I combined the information available in existent metabolic networks published for *L. lactis* (Step 1). The resultant data frame was then combined with the annotated genome sequence (AGS) of *L. lactis* in a new data frame called gene.seq (Step 2). Using the Enzyme Commission (EC) numbers previously available as part of the AGS and the EC numbers collected from the KEGG database, additional equations were collected from the KEGG database and inserted in the data frame gene.seq (Step 3). Finally, potential gaps were detected using the algorithms for tracking metabolites in a metabolic network and for finding potential metabolic gaps. These algorithms were implemented in the R function trackMetab and in the MATLAB function findGaps, respectively. Based on the literature survey, the gaps were filled and the metabolic network was reanalyzed (Step 4).
In the first step of my method (Figure 2.2), I merged net1 and net2, a process that I implemented in the R function MergeNetworks (Appendix 2a). MergeNetworks uses the RXNID as a common identifier to compare the content of the networks net1 and net2. As result, it generates a data frame called net12, which contains the stoichiometric equations available in both networks and two additional columns: correct and fixed. If a stoichiometric equation was equally defined in net1 and net2 (e.g. show exactly the same intermediates and with the same stoichiometric coefficients), the column correct received the value YES and this stoichiometric equation present in net1 was assigned to the column fixed. Otherwise, the value NO was assigned to the column correct and no value was assigned to the column fixed. The rows showing NO in the column correct were manually inspected and the literature was used to confirm the stoichiometry for each of these equations. Then, the literature-derived equation was finally assigned to the column fixed. Ultimately, only the columns RXNID, fixed and subsystem were retained in net12. All other columns were discarded.

In the next step, I merged the data frame net12 and the AGS of L. lactis (AGS) (Table 2.2). This process was implemented in the R function addAGS (Appendix 2b), which uses a unique identifier, in this case RXNID, in the merged data frame, net12, and the column Gene Symbol in the AGS to produce a new data frame called gene.seq. gene.seq contains three columns from AGS, locus, geneid and description; and three columns from net12: net12.geneid, net12.equation and net12.subsystem. One single gene may be expressed as a single enzyme catalyzing multiple reactions. Therefore, some Gene Symbols present in the AGS may be linked to more than one
RXNID present in the data frame net12. In these cases, the value more_than_one is inserted into the corresponding cell of the columns net12.geneid, net12.equation and net12.subsystem, and the information collected from net12 is then placed at the bottom of the data frame gene.seq. As a result, this step generated a new data frame, gene.seq, which is an integration of the data available in net12 and AGS.

The column description of the data frame gene.seq contains the predicted biological functions of genes in AGS, including their EC numbers. Therefore, in the next step the R function getECNumbers (Appendix 2c) isolates these EC numbers and stores them in a new column called ec.numbers in gene.seq.

Table 2.2 – Annotated Genome Sequence of Lactococcus lactis. The annotated genome sequence of Lactococcus lactis contains for each gene: the identification code of its locus, its predicted name, its predicted biological function and its position at the 5’ and 3’ of the DNA sequence. The AGS of L. lactis was downloaded from http://cmr.jcvi.org and contains 2266 genes. This table shows some of the genes present in the AGS of L. lactis.

<table>
<thead>
<tr>
<th>Locus</th>
<th>Gene Symbol</th>
<th>Common Name</th>
<th>5’ End</th>
<th>3’ End</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0274</td>
<td>dnaA</td>
<td>replication initiation protein DnaA</td>
<td>358</td>
<td>1725</td>
</tr>
<tr>
<td>L0275</td>
<td>dnaN</td>
<td>DNA polymerase III, beta chain (EC 2.7.7.7)</td>
<td>1882</td>
<td>3024</td>
</tr>
<tr>
<td>L0252</td>
<td>rexB</td>
<td>subunit B of ATP-dependent exonuclease</td>
<td>3130</td>
<td>6429</td>
</tr>
<tr>
<td>L0251</td>
<td>rexA</td>
<td>subunit A of ATP-dependent exonuclease</td>
<td>6422</td>
<td>10033</td>
</tr>
<tr>
<td>L10283</td>
<td>yabA</td>
<td>transcriptional regulator</td>
<td>10837</td>
<td>10280</td>
</tr>
<tr>
<td>L0161</td>
<td>yyaL</td>
<td>GTP-binding protein</td>
<td>11119</td>
<td>12234</td>
</tr>
</tbody>
</table>

In addition to the EC number, many biochemical databases, such as KEGG and MetaCyc, use the locus ID of each gene as an enzyme ID. As gene.seq contains the locus ID of each predicted gene, I used this information to collect additional reactions playing part in the metabolism of L. lactis. For that, I used the file enzyme_list (Table 2.3), which is available in KEGG database through it FTP server (http://www.kegg.jp/kegg/download/). This file maps gene loci to EC numbers. Thus, I used the locus ID in the integrated network to collect additional EC numbers from enzyme_list. This process was implemented in the getECKEGG (Appendix 2d), which uses the ID of each gene locus (e.g. L0274 for gene
dnaA) present in the data frame gene.seq to search the file enzyme_list for potential EC numbers associated with it. The EC numbers collected are then stored in the column ec.numbers of the data frame gene.seq.

Table 2.3 – Lactococcus lactis’ enzyme_list. The file enzyme_list was obtained from KEGG website and maps gene loci of L. lactis to EC numbers. This table shows part of the data contained in enzyme_list.

<table>
<thead>
<tr>
<th>llactis.locus</th>
<th>ec.list</th>
</tr>
</thead>
<tbody>
<tr>
<td>lla:L0001</td>
<td>ec:5.4.2.6</td>
</tr>
<tr>
<td>lla:L0002</td>
<td>ec:2.7.1.11</td>
</tr>
<tr>
<td>lla:L0003</td>
<td>ec:2.7.1.40</td>
</tr>
<tr>
<td>lla:L0004</td>
<td>ec:1.2.1.12</td>
</tr>
<tr>
<td>lla:L0005</td>
<td>ec:1.2.1.12</td>
</tr>
</tbody>
</table>

At this point, I have collected EC numbers originally in AGS and in the KEGG database. In the next step, I used these EC numbers to collect from KEGG database the actual stoichiometric equations related to each of them. For that, I used another KEGG file called reactions (http://www.kegg.jp/kegg/download/). This file stores a detailed description of each reaction present in KEGG database. It contains the stoichiometric equation of each reaction, the name of the enzyme catalyzing each reaction and its respective EC numbers (Table 2.4). The file reaction was then used to collect the stoichiometric equations related to each EC number present in the column ec.numbers of the data frame gene.seq. This process was implemented in getReactions (Appendix 2e). getReactions collects stoichiometric reactions from KEGG database and compares their configurations in relation to reactions in gene.seq. First, getReactions creates five extra columns in the data frame gene.seq: kegg.equation, kegg.equation.codes, kegg.reaction.id, correct and fixed. Then, for each EC number present in the column ec.number of gene.seq, getReactions copies from the file reactions its respective reaction ID (e.g. R0001) and stoichiometric equations into the columns
kegg.equation, kegg.equation.codes and kegg.reaction.id. The columns correct and fixed receive values as described for MergeNetworks.

Table 2.4 – KEGG’s reactions. The file reactions contains detailed information about each reaction stored in KEGG’s database. This table shows part of the data in the file reactions.

<table>
<thead>
<tr>
<th>ENTRY</th>
<th>R00001</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Polyphosphate polyphosphohydrolase</td>
<td></td>
</tr>
<tr>
<td>DEFINITION</td>
<td>Polyphosphate + n H2O &lt;=&gt; (n+1) Oligophosphate</td>
<td></td>
</tr>
<tr>
<td>EQUATION</td>
<td>C00890 + n C00001 &lt;=&gt; (n+1) C02174</td>
<td></td>
</tr>
<tr>
<td>ENZYME</td>
<td>3.6.1.10</td>
<td></td>
</tr>
</tbody>
</table>

///

<table>
<thead>
<tr>
<th>ENTRY</th>
<th>R00002</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>Reduced ferredoxin:dinitrogen oxidoreductase (ATP-hydrolysing)</td>
<td></td>
</tr>
<tr>
<td>DEFINITION</td>
<td>16 ATP + 16 H2O + 8 Reduced ferredoxin &lt;=&gt; 8 e- + 16 Orthophosphate + 16 ADP + 8 Oxidized ferredoxin</td>
<td></td>
</tr>
<tr>
<td>EQUATION</td>
<td>16 C00002 + 16 C00001 + 8 C00138 &lt;=&gt; 8 C05359 + 16 C00009 + 16 C00008 + 8 C00139</td>
<td></td>
</tr>
<tr>
<td>COMMENT</td>
<td>a part of multi-step reaction (see R05185, R00002+R000067+R00153+R02802+R04782)</td>
<td></td>
</tr>
<tr>
<td>RPAIR</td>
<td>RP: RP00003  C00002_C00008 main</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP: RP00010  C00002_C00009 leave</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RP: RP05676  C00001_C00009 leave</td>
<td></td>
</tr>
<tr>
<td>ENZYME</td>
<td>1.18.6.1</td>
<td></td>
</tr>
</tbody>
</table>

The data frame gene.seq constructed as above contains the stoichiometric equations present in existing metabolic networks, in this case for *L. lactis*, and all reactions from KEGG database that are associated with the locus IDs and/or EC numbers present in the AGS of this organism. As a result, I collected more than 3000 reactions potentially involved in the
metabolism of *L. lactis*. However, for reconstructing a high-quality metabolic network, one must include preferentially reactions that show evidence of participation in the metabolism of the organism under study. Therefore, in the next step I used the literature and the web databases KEGG, Brenda, and MetaCyc to filter these reactions selecting only the ones showing biological evidence of involvement in the metabolism of *L. lactis*. The criteria used for filtering these reactions may change according to the final application of the metabolic network (Thiele and Palsson, 2010). The list of reactions resultant from this process was considered the first draft of the new metabolic network of *L. lactis*.

In the next step I then searched for potential gaps in the first draft of the metabolic network of *L. lactis*. This was achieved by using a new algorithm I developed (see section 2.2). This algorithm was implemented in the MATLAB function `findGaps`, which uses COBRA Toolbox (Thiele and Palsson, 2010) within MATLAB to perform FBA and identify reactions and metabolites involved in metabolic gaps. COBRA Toolbox requires a metabolic network in Systems Biology Markup Language (SBML) format, the standard language used to represent biological processes (Hucka et al., 2003). Therefore, it is necessary to convert the metabolic network into SBML format before using `findGaps`. Since there was no freely available software for converting CSV files into SBML files, I wrote two functions – one for R and the other for MATLAB – which, when coupled, perform this file format conversion.

**CSV to SBML format**

Firstly, I reorganized the metabolic network in a unique XLS file (Table 2.1 – Appendix) containing 5 worksheets: RXNS, METS, COMPS, MODEL and GENES. RXNS is the main spreadsheet. It contained all the stoichiometric equations with their IDs, the compartment where each reaction occurs, the reactions that will be part of the objective function, potential association between genes, the EC number of each reaction and, finally, the upper and lower bounds of each reaction. The worksheet METS described all the compounds involved in the metabolic network while COMPS described all the cellular compartments. *L. lactis* is a prokaryote and, thus, has no intracellular compartments. All the intracellular reactions occur in the cytosol, which is identified by the letter c. However, in order to have more control in modeling the cell’s metabolism, I created two extra virtual spaces, namely compartments e and b. Compartment e contained equations representing transport of metabolites across the
membrane, while compartment b contained exchange equations, defining which compounds are available for compartment e. In other words, compartment b defined the composition of the culture medium while compartment e defined the rates at which the bacteria uptake compounds from the medium. Finally, the MODEL worksheet contained a general description of the network, while the GENES worksheet contained additional information to be potentially added to each gene.

Once the XLS file was created and organized as above, its conversion into SBML was achieved in two steps. First, I converted each worksheet within the XLS file into individual CSV files. This process was implemented in the R function XLStoCSV (Appendix 2f), which loads the XLS file into an R session and saves each of its worksheets into different CSV files. Then, I used each CSV file to compile a unique SBML file; a process that was implemented in the MATLAB function makeSBML (Appendix 2g). It loads each of the CSV files previously generated by R and converts it into a unique SBML file. This SBML file was then ready to be uploaded to the COBRA Toolbox (see Becker et al., 2007 for details about uploading SBML files into COBRA Toolbox).

With the SBML ready and loaded to COBRA Toolbox, the next step was to find the metabolic gaps using my MATLAB function findGaps (Appendix 2h). The metabolic network must contain no flux constraint, which is achieved by simply leaving lower and upper bounds of each reaction as default. findGaps then optimizes the network for maximizing each reaction, one at a time, and saves the results of each simulation in a variable called fluxes (Table 2.5). The network is optimized by simulating its fluxes when setting different reactions as objective equations, one at a time. Every reaction that shows zero flux in the variable fluxes is potentially involved in metabolic gaps. In order to identify which intermediate of these reactions is responsible for the metabolic gap, findGaps then performs a set of simulations for maximizing each reaction showing zero flux in the variable fluxes. For example, if a reaction involved in a metabolic gap has four intermediates, findGaps then performs four new simulations removing one specific intermediate at a time. If the flux resultant from a simulation changes to a value that differs from zero, the metabolite removed in this specific simulation is likely to be accumulating inside the cell or it was not being produced by any other reaction present in the model. The result of each simulation is stored in a variable called gaps (Table 2.6), which contains three columns: the reaction id (reaction id), the name of the intermediate being evaluated (compound
name) and the result of the analysis (result). The result column receives one of two possible values: no_change, if removing an intermediate produced no change in flux, or check, if removing an intermediate produced a non-zero flux (i.e. the evaluated intermediate was potentially involved in metabolic gaps and had to be inspected). This algorithm considerably facilitated the process of gap finding.

Table 2.5. Variable fluxes. The function findGaps optimizes the network for maximizing each reaction, one at a time, and saves the results of each simulation in a variable called fluxes.

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Flux</th>
</tr>
</thead>
<tbody>
<tr>
<td>'accA_1'</td>
<td>1000</td>
</tr>
<tr>
<td>'accB_1'</td>
<td>-1000</td>
</tr>
<tr>
<td>'accD_1'</td>
<td>195.2653062</td>
</tr>
<tr>
<td>'ackA1_1'</td>
<td>1000</td>
</tr>
<tr>
<td>'ackA2_1'</td>
<td>1000</td>
</tr>
<tr>
<td>'acpD_1'</td>
<td>1000</td>
</tr>
<tr>
<td>'acpD_1_dup'</td>
<td>-1000</td>
</tr>
<tr>
<td>'acpS_1'</td>
<td>0.030469953</td>
</tr>
<tr>
<td>'acpS_1_dup'</td>
<td>-0.030469953</td>
</tr>
<tr>
<td>'add_1'</td>
<td>0</td>
</tr>
<tr>
<td>'add_2'</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 2.6 Variable gaps. The function *findGaps* generates a variable called *gaps*, which contains the results of evaluations performed to identify metabolites involved in metabolic gaps. Every metabolite showing “Check” in the column *result* is potentially involved in metabolic gaps.

<table>
<thead>
<tr>
<th>reaction id</th>
<th>compound name</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>'argB_1'</td>
<td>'ADP-C10H15N5O10P2'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argB_1'</td>
<td>'ATP-C10H16N5O13P3'</td>
<td>'no_change'</td>
</tr>
<tr>
<td></td>
<td>'N=acetyl=5=glutamyl=phosphate-C7H12NO8P'</td>
<td>'Check'</td>
</tr>
<tr>
<td>'argB_1'</td>
<td>'N=acetyl=L=glutamate-C7H11NO5'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argC_1'</td>
<td>'H(+)\text{-}H'</td>
<td>'no_change'</td>
</tr>
<tr>
<td></td>
<td>'N=acetyl=5=glutamyl=phosphate-C7H12NO8P'</td>
<td>'no_change'</td>
</tr>
<tr>
<td></td>
<td>'N=acetyl=L=glutamate=5=semialdehyde-C7H11NO4'</td>
<td>'Check'</td>
</tr>
<tr>
<td>'argC_1'</td>
<td>'NADP(+)\text{-}C21H29N7O17P3'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argC_1'</td>
<td>'NADPH-C21H30N7O17P3'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argC_1'</td>
<td>'phosphate-H3PO4'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argD_1'</td>
<td>'2=oxoglutarate-C5H6O5'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argD_1'</td>
<td>'L=glutamate-C5H9NO4'</td>
<td>'no_change'</td>
</tr>
<tr>
<td>'argD_1'</td>
<td>'N2=acetyl=L=ornithine-C7H14N2O3'</td>
<td>'Check'</td>
</tr>
<tr>
<td></td>
<td>'N=acetyl=L=glutamate=5=semialdehyde-C7H11NO4'</td>
<td>'no_change'</td>
</tr>
</tbody>
</table>

Once potential metabolic gaps were identified in the metabolic network of *L. lactis*, I then inspected the network to understand how each metabolic gap could be filled. I developed an *R* function, *track.metab* (Appendix 2i), which allows users to track a metabolic pathway consuming or producing a metabolite involved in metabolic gaps. *track.metab* is an
interactive function that searches for metabolites in a metabolic network and presents to the user a dialog box containing all the equations involving the specified metabolite (Figure 2.3). By analyzing this list of equations, the user can then easily verify if the respective compound is being produced and/or consumed by any of the presented equations. Alternatively, the user may select one of the presented equations to track a different metabolite. The equation selected by the user is then split and its intermediates are presented to the user in a new dialog box. The user then selects a compound from the list and a new dialog box will present all the reactions involving this specific compound, which restarts the cycle. This tool allowed quick tracking of metabolites in the genome-scale metabolic network of *L. lactis*. Using `track.metab` I found the missing steps in the production or consumption of metabolites. Then, using the literature and web biochemical databases I filled the gaps by adding missing reactions, removing existing reactions or changing the stoichiometry of existing reactions. The process of finding and filling gaps was repeated until no further gaps were identified.

![Figure 2.3. R function track.metab.](image)

**Figure 2.3. R function track.metab.** The R function `track.metab` allows users to track metabolites in a metabolic network stored in CSV format. It tracks the desired compound and presents to the user every reaction involving this compound. Selecting one of the presented reactions, `track.metab` then split the reaction in intermediates and present all intermediates to the user. Selecting an intermediate, the user restarts the cycle and every reaction involving this intermediate will be presented in a new dialog box.
2.4. Results and discussion

Here I presented a method for merging and expanding existing genome-scale metabolic networks. It is achieved by comparing the configuration of stoichiometric equations between networks and reporting reactions requiring manual verification. The use of the annotated genome sequence and KEGG database, allows obtaining additional reactions potentially related to the organism under study, which may result in additional knowledge.

The step one of the reconstruction method presented here involves the merging of metabolic networks by the use of a common reaction identifier between these networks (see section 2.3). Although genome-scale modeling is becoming more popular, there is not yet a clear standardization to be used in reconstruction procedures. In 2008 (Herrgard et al., 2008), a discussion about a potential consensus for the reconstruction of yeast metabolic networks was initiated. Since then, protocols such as the one produced by Thiele and Palsson (2010) have improved standardization between newly reconstructed networks. However, there is still a need for well-defined reconstruction rules allowing easy merging of metabolic networks reconstructed by different research groups. Therefore, when merging metabolic networks, different reaction identifiers may be used in step one according to the reconstruction method originally used.

The method developed here for finding metabolic gaps, findGaps, was able to generate accurate reports about reactions and metabolites potentially involved in gaps. As it tests one metabolite at a time, this method is limited by the number of metabolites from the same reaction that may be involved in gaps. If two or more metabolites from the same reactions are not produced or consumed by further reactions, they will not be reported by findGaps. However, by filling the gaps reported and reapplying findGaps, the number of reactions involving multiple gaps decreases considerably and most of the gaps can be found.
The R function `track.metab` was developed to track metabolites in a genome-scale metabolic network. This function allows to quickly verifying the production or consumption of a specific compound, which, coupled to `findGaps`, considerably reduces the time required for gap finding and filling.

Using the method and computer tools described above (section 2.3), a new genome-scale metabolic network was generated for *L. lactis* based on existing metabolic networks. It contains 921 reactions (Appendix 2j) (Figure 2.4), where 154 are transport reactions, 563 are intracellular reactions and 204 are exchange reactions between virtual compartments. This metabolic network contains 90 additional reactions than the previous metabolic network published by Oddone et al. 2009, and 96 more reactions than Oliveira et al. 2005 (Table 2.7). About 20 of these are reactions involving the transport of metabolites from one cellular compartment to another, while the rest are reactions occurring within the intracellular compartment. This new metabolic network combines the information available in existing networks and additional reactions obtained from KEGG database. The method developed for finding metabolic gaps was able to quickly report reactions and metabolites involved in gaps. Most of these gaps were filled by additional reactions or by changing the reversibility of existing reactions based on information available in the literature and chemical databases (i.e. KEGG, MetaCyc and Brenda).

**Table 2.7. Network features.** This table shows the number of reactions in existent metabolic networks and in the integrated metabolic network produced here.

<table>
<thead>
<tr>
<th>Metabolic models</th>
<th>Transport</th>
<th>Intracellular</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oliveira et al., 2005</td>
<td>129</td>
<td>492</td>
<td>621</td>
</tr>
<tr>
<td>Oddone et al., 2009</td>
<td>132</td>
<td>495</td>
<td>627</td>
</tr>
<tr>
<td>Integrated network</td>
<td>154</td>
<td>563</td>
<td>717</td>
</tr>
</tbody>
</table>
The genome-scale metabolic network of *L. lactis* reconstructed in this chapter shows the existence of some key compounds functioning as network hubs (Figure 2.5). These compounds are connected to many different reactions and are considered essential to maintaining the structure of the metabolic network (Zhu et al., 2007). The existence of network hubs is a common characteristic feature of most biological networks and usually provides robustness to the system (Zhu et al., 2007). Generally, metabolic hubs are related to essential functions of the cell, such as electron transport and energy generation, and reactions involving these compounds tend to be more conserved, as showed by Nielsen (2003). Nielsen identified essentially the same compounds functioning as metabolic hubs in the genome-scale metabolic networks of *Escherichia coli* (gram-negative bacteria), *Saccharomyces cerevisiae* (yeast), *Helicobacter pylori* (gram-negative bacteria) and *Haemophilus influenzae* (gram-negative bacteria), which demonstrated how conserved the central carbon metabolism is across different species. I analyzed the metabolic network reconstructed using my method and, not surprisingly, I identified the same main hubs described by Nielsen (2003). Compounds from the energy metabolism, such as ATP, ADP, NAD⁺, NADH, H⁺, CO₂ and pyruvate are the compounds showing the highest number of connections (Figure 2.5). Metabolic hubs were identified by calculating the number of reactions associated to each compound in the metabolic network. Hubs were considered those metabolites associated to several different metabolic reactions.
Figure 2.4. Integrated metabolic network of \textit{Lactococcus lactis}. This figure shows the metabolic network of \textit{Lactococcus lactis} reconstructed from two existent metabolic networks published by Oliveira et al., (2005) and Oddone et al., (2009). This figure was generated using the software Cytoscape (Shannon et al., 2003).
Figure 2.5. Hubs of the integrated metabolic network. This figure shows the compounds of the integrated metabolic network that are involved in the highest number of reactions.

The reconstruction method described here and its implementation in computational tools was essential both for saving time and for reducing the high possibility of human error in the integration of previous metabolic networks of *L. lactis*. The algorithms developed for merging and comparing metabolic networks enabled quick visualization of differences between networks, while other algorithms allowed me to collect additional reactions from the KEGG database. Furthermore, the algorithms for gap finding and tracking metabolites in metabolic networks promoted quick identification and correction of metabolic gaps. The function `track.metab`, in particular, allows users to “visualize” the pathways followed by each compound, which ultimately allows for a better understanding of the metabolic network.
and, consequently, the metabolism of the organism under study. The genome-scale metabolic network of *L. lactis* presented in this chapter was reconstructed mainly to illustrate the use of new methods for reconstructing and curating metabolic networks. Additional experiments, simulations and tuning are necessary to quantitatively validate this metabolic network.

### 2.5. Conclusion

In this chapter, I presented a new method for merging, expanding and finding metabolic gaps in existing genome-scale metabolic networks. Using this method, a new genome-scale metabolic network was reconstructed for *L. lactis*, containing reactions originated from existing networks and additional reactions from KEGG database. Furthermore, I semi-automated this method by implementing each of its steps in computer functions developed in R and MATLAB, which considerably reduces the time-spent in the reconstruction process and decreases the chance of potential human errors. The next step I believe would be implementing the computer tools presented here in a single package.
Chapter 3. MetaBox - a new algorithm for deconvolution and identification of metabolites analyzed by GC-MS.

3.1. Introduction

Metabolomics is amongst the newest omics-related technologies and is already considered essential in systems biology studies (Patti et al., 2012). It is capable of screening large numbers of low molecular mass compounds in biological samples and has been recently applied for validating and expanding genome-scale metabolic models (Thiele and Palsson, 2010; Feist et al., 2009) using analytical techniques such as gas chromatography–mass spectrometry (GC-MS). GC-MS is currently one of the most commonly used techniques in metabolomics analysis as it is able to separate complex mixtures of metabolites with high efficiency (Villas-Bôas et al., 2011).

The chromatograms generated by GC-MS can be analyzed by software such as the Automated Mass Spectral Deconvolution and Identification System (AMDIS), distributed by the National Institute of Standards and Technology (NIST). AMDIS was originally developed in 1999 for the identification of chemical weapons and related compounds in complex chemical mixtures (Stein, 1999). Since then, AMDIS has been used in environmental chemistry (Finck et al. 2004; Furtula et al. 2006) and, more recently, in metabolomics studies (Hamzehzarghani et al. 2008; Carneiro et al. 2010). Today, AMDIS is among the most commonly used software for deconvoluting chromatograms generated by GC-MS and for identifying and quantifying metabolites in biological samples (Aggio et al. 2011; see section 1.2.2 for details about deconvolution).

While AMDIS is good for identifying target metabolites in biological samples, it also has a number of limitations when applied for metabolomics. First, it uses a different mass fragment as reference when quantifying the same compound in different samples (Aggio et al. 2011), limiting its usage for comparing the intensity of compounds across samples or experimental conditions, the most common approach in metabolomics. Second, it generates a high rate of false positives (Behrends et al. 2011), requiring considerable manual curation and, ultimately, may mislead the final biological interpretation. And finally, it generates results in a layout that does not make the data amenable to further processing and analysis (Smart et al., 2010). As a result, the data sets produced by AMDIS require manual inspection and corrections that may take months to be completed depending on the size of the data set. With the exponential growth of metabolomics and the consequent explosion in the amount of GC-MS-generated data (Dettmer et al. 2007; Patti et al. 2012), AMDIS is becoming incompatible with such demands; therefore, an improved algorithm to identify and quantify metabolites analyzed by
GC-MS and able to report results is a format that facilitates further data processing is urgently needed.

In the last couple of years, many programs (Aggio et al. 2010; Aggio et al. 2011, Styczynski et al. 2007, Xia and Wishart, 2011, Choe et al. 2012) have been developed to automate processes for the analysis of metabolomics data. These tools allow for quick data normalization, statistical analysis and the production of graphs for data visualization. For example, Metab, an R (R Development Core Team, 2012) package I developed in 2011 (Aggio et al. 2011), is able to automatically correct the abundances of metabolites previously identified by AMDIS and to combine AMDIS’ results in a single spreadsheet, which considerably facilitates further data processing steps. Metab also performs data normalization (e.g. internal standard normalization, biomass normalization and uncultured medium normalization) and basic statistical analyses. However, to the best of my knowledge, there are no freely available metabolomics tools capable of deconvoluting and identifying metabolites. Thus, despite its limitations, AMDIS remains the most popular software in metabolomics.

In this chapter, I present a new algorithm (MetaBox) to deconvolute, identify and quantify compounds in biological samples analyzed by GC-MS. MetaBox makes use of a mass spectral library and a set of procedures to determine whether compounds in the spectral library are present in biological samples. As a result, MetaBox generates a list of metabolites identified and their respective intensities in different analyzed samples. Its primary objective is to reduce the effort spent in processing data sets generated from metabolomics studies.

In order to validate the results generated by MetaBox, I compare its identification and quantification performances against results produced by AMDIS and Metab. As MetaBox is considerably time consuming if performed manually, I implemented this algorithm in a new R package also called MetaBox, which considerably facilitates its usage. The MetaBox package automates the deconvolution and identification of metabolites analyzed by GC-MS and also incorporates Metab’s functions for normalization and data processing. In addition, MetaBox uses pop-up dialog boxes, which makes it more accessible to novice R users.
3.2. MetaBox – The algorithm

MetaBox is an algorithm that makes use of a spectral library and different procedures to deconvolute, identify and quantify metabolites in biological samples.

Spectral library

The spectral library (Table 3.1) used by MetaBox is similar to the spectral library used by AMDIS. It consists of a table containing the features of standard metabolites used for identification. Standard metabolites are compounds of known identity analyzed, in this case using GC-MS, to detect their retention times (RTs) and fragmentation patterns (spectra). This information is stored and further used to identify metabolites in biological samples. For this, the spectra from a new sample are cross-referenced with those in the library and, when a match is found using RTs, the metabolite is identified. Spectra library can be built for a specific experiment, although it is generally built to be used as reference for most of the experiments performed in a specific metabolomics laboratory.

The spectral library used by MetaBox is a table containing the name of each standard metabolite in the first column, its expected RTs in the second column, the $m/z$ of the four most abundant ion mass fragments (IMF) in the next four columns, and the intensity ratios between these fragments in the last three columns. The RT associated to each metabolite is expected to be considerably reproducible, with a RT shift of about 0.2 to 0.5 minutes. The intensity ratios between fragments are a representation of the fragmentation pattern associated with a metabolite. The three or four most abundant IMFs are the ones showing the highest intensities and, thus, show less interference of a potential background noise. For example, the row of the spectral library representing the metabolite alanine has the following information: Alanine (compound name), 10.975 (RT), 102 ($m/z$ of M1 or the most abundant IMF), 59 ($m/z$ of M2 or the second most abundant IMF), 58 ($m/z$ of M3 or the third most abundant IMF), 70 ($m/z$ of M4 or the fourth most abundant IMF), 0.184 (the intensity of M2 divided by the intensity of M1), 0.146 (the intensity of M3 divided by the intensity of M1) and 0.105 (the intensity of M4 divided by the intensity of M1) (Table 3.1 and Figure 3.1A).
Table 3.1. Spectral library. The spectral library used by MetaBox contains the name of standard compounds, their respective expected RTs, the mass-to-charge ratio of the most abundant ion mass fragments (M1, M2, M3 and M4) and the intensity ratios between M1, M2, M3 and M4. This table shows examples of entries of the spectral library.

<table>
<thead>
<tr>
<th>Compound name</th>
<th>RT (min)</th>
<th>Most abundant ion mass fragments (m/z)</th>
<th>Ratios relative to M1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>M1</td>
<td>M2</td>
</tr>
<tr>
<td>2,3-Butanediol</td>
<td>5.626</td>
<td>45</td>
<td>57</td>
</tr>
<tr>
<td>2-Oxobutyric acid</td>
<td>5.702</td>
<td>57</td>
<td>59</td>
</tr>
<tr>
<td>Oxalic acid</td>
<td>5.892</td>
<td>59</td>
<td>45</td>
</tr>
<tr>
<td>C6:0 acid (Capronic acid)</td>
<td>6.185</td>
<td>74</td>
<td>87</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid</td>
<td>6.439</td>
<td>43</td>
<td>74</td>
</tr>
<tr>
<td>2-Oxovaleric acid</td>
<td>7.167</td>
<td>71</td>
<td>43</td>
</tr>
<tr>
<td>Malonic acid</td>
<td>7.362</td>
<td>101</td>
<td>59</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid</td>
<td>7.689</td>
<td>57</td>
<td>85</td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid</td>
<td>7.787</td>
<td>57</td>
<td>85</td>
</tr>
<tr>
<td>Fumaric acid</td>
<td>8.299</td>
<td>113</td>
<td>85</td>
</tr>
<tr>
<td>Levulinic acid</td>
<td>8.607</td>
<td>43</td>
<td>99</td>
</tr>
<tr>
<td>Succinic acid</td>
<td>8.955</td>
<td>115</td>
<td>55</td>
</tr>
<tr>
<td>Lactic acid</td>
<td>9.103</td>
<td>59</td>
<td>103</td>
</tr>
<tr>
<td>C8:0 acid (Caprylic acid)</td>
<td>9.354</td>
<td>74</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>------</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Benzoic acid</td>
<td>9.492</td>
<td>105</td>
<td>77</td>
</tr>
<tr>
<td>Dodecane</td>
<td>9.575</td>
<td>57</td>
<td>43</td>
</tr>
<tr>
<td>1-Phenylethanol</td>
<td>9.837</td>
<td>107</td>
<td>79</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>10.069</td>
<td>127</td>
<td>59</td>
</tr>
<tr>
<td>Citraconic acid</td>
<td>10.131</td>
<td>127</td>
<td>59</td>
</tr>
<tr>
<td>2-Hydroxybutyric acid</td>
<td>10.3</td>
<td>117</td>
<td>73</td>
</tr>
<tr>
<td>Citramalic acid</td>
<td>10.578</td>
<td>117</td>
<td>43</td>
</tr>
<tr>
<td>Nicotinic acid</td>
<td>10.696</td>
<td>106</td>
<td>78</td>
</tr>
<tr>
<td>d4-Alanine</td>
<td>10.947</td>
<td>106</td>
<td>62</td>
</tr>
<tr>
<td>Alanine</td>
<td>10.975</td>
<td>102</td>
<td>59</td>
</tr>
</tbody>
</table>

*Deconvoluting, identifying and quantifying*

GC-MS machines generate a file for each analyzed sample. This file contains a list of spectra and RTs. MetaBox then searches GC-MS files for each metabolite contained in the spectral library. Based on the RT and fragmentation pattern of a given metabolite, MetaBox uses four different stages to determine whether it is present in a sample analyzed by GC-MS (Figure 3.1).
Figure 3.1. MetaBox - Deconvolution and identification algorithm. The algorithm for deconvolution and identification of metabolites analyzed by gas chromatography – mass spectrometry (GC-MS) searches GC-MS files for every single compound contained in the mass spectral library. (A) It makes use of the expected retention time (RT) for searching potential chromatographic peaks representing a metabolite contained in the spectral library. Then, it (B) verifies if the ion mass fragments expected to originate from this specific metabolite were detected at the same RT, (C) at the expected intensities and (D) if these ion mass fragments show positive correlation between their intensities. Doing so, it determines if a compound is present in a biological sample. If so, it registers the intensity of the compound’s most abundant mass fragment (M1) as an indication of its abundance. See Deconvoluting and identifying for more details.
MetaBox starts the process by collecting from the spectral library the expected RT of a metabolite, such as alanine. It then uses the expected RT to trim the chromatogram, keeping only a RT window in which the IMFs generated by alanine are likely to be found. MetaBox then selects the region of the chromatogram contained in the time range of the expected RT ± w, where w is window parameter in the algorithm and given in minutes. For example, alanine has an expected RT of 10.975 minutes. If window = 1, MetaBox searches for alanine 9.975 min to 11.975 min region of the chromatogram (Figure 3.1A). Every peak in a GC chromatogram represents an analyte, which could be a metabolite, a derivative product of a metabolite or an impurity. Therefore, within the chromatogram region previously selected, MetaBox searches for every potential peak of the mass fragment M1. For Alanine, M1 is the IMF 102 m/z (Figure 3.1A). Thus, MetaBox searches for potential peaks of the IMF 102 m/z within the RT window 9.975 min to 11.975 min. The RT of each potential peak is then stored in a variable named RTList and analyzed in subsequent stages. RTList represents every potential peak in the chromatogram that could have been produced by a specific metabolite, such as alanine. MetaBox then uses four different stages and a system of points to identify which RT in the RTList is most likely to represent a metabolite, in this case alanine. Several metabolites can generate similar m/z’s. However, the RT and the proportion between m/z’s generated by different metabolites are expected to be different.

**Stage 1 – Finding IMFs**

The IMFs generated by the same analyte are expected to reach the recorder of the GC-MS approximately at the same time. Consequently, IMFs originating from the same analyte are likely to be present at the same RT of the chromatogram. MetaBox then searches the chromatogram for the presence of M2, M3 and M4 at each RT previously stored in the RTList. When an IMF is found, the respective stored RT then receives a 1-point score. As a result, every RT in the RTList containing the four IMFs representing the known compound (e.g. alanine) receives a score of 3 (Figure 3.1A). Further confirmation of the relative m/z ratios of M2, M3 and M4, respectively, to the m/z of M1, is performed in the subsequent three stages (Stages 2 – 4), described below.

**Stage 2 – Checking IMFs peaks**

Every IMF originating from the same analyte is expected to have its highest intensity peak approximately at the same RT (Figure 3.1B). Therefore, in this second stage, MetaBox searches for potential peaks in the intensities of M2, M3 and M4, and verifies whether the RT
of these peaks coincides with any RT previously stored in RTList. For each positive match, the respective stored RT receives 1 additional point score. At the end of the second stage, a stored RT that coincides with the peaks of all four IMFs originating from the same compound would end up receiving a score of 6: three points collected from the first stage and three collected from the second.

**Stage 3 – Checking fragmentation pattern**

For each RT in RTList, MetaBox verifies if the intensities of M1, M2, M3 and M4 match with their respective expected ratios stored in the spectral library (Figure 3.1C). For example, the IMFs M2 (59 m/z), M3 (58 m/z) and M4 (70 m/z) from alanine are expected to have ratios of 0.184, 0.146, and 0.105, respectively, relative to the IMF M1 (102 m/z). Each RT in RTList then receives one extra point score for each IMF showing the expected ratio in relation to M1. However, the observed and expected ratios do not always equate. Therefore, it is necessary to define a similarity threshold to determine how similar the observed and expected ratios need to be in order to consider an IMF a positive match. For this, I created a parameter called *match factor*, which indicates the minimum proportion (or percentage) the observed and expected ratios need to be so as to consider them similar. The *match factor* receives values from 0 to 99, where 99 means a threshold of 99% similarity between the observed and expected ratios to consider an IMF a positive match. The match factor is calculated by simply dividing the lower value between observed and expected ratios by the highest value between observed and expected ratios. For example, if the observed ratio between two IMFs is 100 and the expected ratio between these IMFs is 70, the match factor is calculated by dividing 70 over 100. And vice versa, if the observed ratio is 70 and the expected ratio is 100, then the match factor is calculated, again, by dividing 70 by 100. Every positive match results in an extra point for its respective RT in the RTList.

**Stage 4 – Checking the correlation between IMFs**

IMFs originating from the same metabolite are expected to have their intensities increasing and decreasing approximately at the same RT. Thus, the intensities of these IMFs are expected to be positively correlated (Figure 3.2). Pearson’s correlation coefficient is then used to estimate the strength of the association between the intensities of M1 and M2, M1 and M3, and M1 and M4 in each stored RT ± 0.5 min (Figure 3.1D). Each stored RT in the RTList then receives one extra point for each IMF showing association with M1. The parameter *cor* defines how strong this association must be to be accepted. It receives values
from 0 to 1, where 1 means that a perfect positive correlation is required to generate an extra point to its respective RT.

Figure 3.2. Correlation between mass fragments. The intensities of mass fragments originated from the same analyte are expected to present strong correlation. Their intensities increase and decrease at about the same retention time. Therefore, MetaBox uses Pearson's correlation ($r$) to validate if fragments belong to the same analyte.

Finally, every RT in the RTList is ranked according to the number of point scores received. The parameter $score$ defines the minimum score required for an RT to be considered as representing a real analyte. RTs showing a lower score than the value defined in $score$ are permanently removed from the analysis. The RT showing the highest score is then selected and the intensity of M1 in this specific RT is collected and stored. This intensity represents the abundance of this specific analyte in the sample. For each compound present in the spectral library, MetaBox always registers the intensity of M1 for every sample. If more than one stored RT received equal highest score, the RT having the lowest difference between expected and obtained RT is shortlisted.

The entire process described in this section is repeated for each compound present in the spectral library and for each file generated from GC-MS. As a result, MetaBox generates a table containing the names of the identified metabolites in the first column and the intensities of their most abundant mass fragment (M1), as defined in the spectral library, contained in each analyzed GC-MS file in the following columns (Table 3.2).
Table 3.2. MetaBox’ result. MetaBox generates a table containing the name of identified compounds in the first column and the intensity of their most abundant mass fragment in each analyzed GC-MS file in the following columns. This table shows an example of results generated by MetaBox. Samples 1 to 3 originated from a hypothetical experimental condition 1, while Samples 4 to 6 originated from a hypothetical experimental condition 2.

<table>
<thead>
<tr>
<th>Compound name</th>
<th>Sample 1</th>
<th>Sample 2</th>
<th>Sample 3</th>
<th>Sample 4</th>
<th>Sample 5</th>
<th>Sample 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Oxovaleric acid</td>
<td>64510</td>
<td>57774</td>
<td>51912</td>
<td>3730641</td>
<td>22307018</td>
<td>17489020</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>4564992</td>
<td>4644352</td>
<td>6667776</td>
<td>27148745</td>
<td>7466540</td>
<td>6548544</td>
</tr>
<tr>
<td>Alanine</td>
<td>5564416</td>
<td>5217164</td>
<td>45408</td>
<td>417531283</td>
<td>477023047</td>
<td>407586597</td>
</tr>
<tr>
<td>beta-Alanine</td>
<td>3563234</td>
<td>3755520</td>
<td>2350592</td>
<td>247861376</td>
<td>281566205</td>
<td>258023392</td>
</tr>
<tr>
<td>Citric acid</td>
<td>2592256</td>
<td>2515968</td>
<td>6881280</td>
<td>213416490</td>
<td>116494749</td>
<td>225624501</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>916352</td>
<td>912192</td>
<td>938351</td>
<td>28508918</td>
<td>52309453</td>
<td>39220249</td>
</tr>
<tr>
<td>N2-Acetyl-L-lysine</td>
<td>3299</td>
<td>3324</td>
<td>3853</td>
<td>36478</td>
<td>24719</td>
<td>60541</td>
</tr>
<tr>
<td>Succinic acid</td>
<td>8224343</td>
<td>4596736</td>
<td>8387255</td>
<td>751635984</td>
<td>591122942</td>
<td>546476776</td>
</tr>
<tr>
<td>Threonine</td>
<td>3454551</td>
<td>4549632</td>
<td>3203255</td>
<td>285851977</td>
<td>207422622</td>
<td>239017976</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>7014912</td>
<td>7076178</td>
<td>5636096</td>
<td>105668435</td>
<td>91527480</td>
<td>102433475</td>
</tr>
<tr>
<td>Vanillic acid</td>
<td>2242090</td>
<td>2246144</td>
<td>3068416</td>
<td>52282663</td>
<td>42274639</td>
<td>43798071</td>
</tr>
</tbody>
</table>

3.3. Validation

In order to validate MetaBox, I compared its performance against AMDIS and Metab in identifying and quantifying metabolites analyzed by GC-MS.

3.3.1 Method

Two standard mixtures containing 15 metabolites with known concentrations were prepared (Table 3.3). Each standard sample was divided into three aliquots, or triplicates (Figure 3.5). Twenty µl of internal standard, 2,3,3,3,-d4 alanine (10mM), was added to each replicate prior to derivatization using methyl chloroformate (MCF) (Smart et al., 2010). As the same amount of internal standard is added to each replicate, the final concentration of internal standard is
expected to be the same in every sample. Each derivatized sample was then analyzed by the GC-MS Agilent GC7890 coupled to an MSD5975 (Agilent technologies), with a quadrupole mass selective detector (EI) operated at 70 eV and a column ZB-1701 (Phenomenex), 30 m x 250 m (internal diameter) x 0.15 (film thickness), with 5 m guard column and operated in scan mode (start after 6 min; mass range 38-650 a.m.u. at 1.47 scans/s) (Smart et al. 2010).

Table 3.3. Two standard samples for MetaBox validation.

<table>
<thead>
<tr>
<th>Metabolites</th>
<th>Mix 1 Concentration (mmol/L)</th>
<th>Mix 2 Concentration (mmol/L)</th>
</tr>
</thead>
<tbody>
<tr>
<td>N-acetyl-L-lysine</td>
<td>1.625</td>
<td>3.250</td>
</tr>
<tr>
<td>L-Norvaline</td>
<td>3.250</td>
<td>3.250</td>
</tr>
<tr>
<td>2-Oxovaleric acid</td>
<td>1.625</td>
<td>3.250</td>
</tr>
<tr>
<td>3-Hydroxyoctanoic acid</td>
<td>3.250</td>
<td>3.250</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>1.625</td>
<td>3.250</td>
</tr>
<tr>
<td>Alanine</td>
<td>1.625</td>
<td>1.625</td>
</tr>
<tr>
<td>ß-Alanine</td>
<td>1.625</td>
<td>0.812</td>
</tr>
<tr>
<td>Capric acid</td>
<td>3.250</td>
<td>3.250</td>
</tr>
<tr>
<td>Citric acid</td>
<td>0.812</td>
<td>1.625</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>0.812</td>
<td>0.812</td>
</tr>
<tr>
<td>Succinic acid</td>
<td>3.250</td>
<td>3.250</td>
</tr>
<tr>
<td>Threonine</td>
<td>1.625</td>
<td>1.625</td>
</tr>
<tr>
<td>Tryptophan</td>
<td>1.625</td>
<td>1.625</td>
</tr>
<tr>
<td>2-Hydroxycinnamic acid</td>
<td>0.812</td>
<td>1.625</td>
</tr>
<tr>
<td>Vanillic acid</td>
<td>0.812</td>
<td>1.625</td>
</tr>
</tbody>
</table>
Figure 3.3. **Experimental design for MetaBox validation.** Two standard samples were prepared and divided in triplicates. Each replicate was derivatized using MCF and analyzed by GC-MS (Agilent).

The 6 GC-MS files (Figure 3.3) generated as above were then divided in two conditions: *Mix 1*, which includes all GC-MS files originating from the standard mix 1, and *Mix 2*, which contains all GC-MS files originating from the standard mix 2. Each single sample was subsequently processed by AMDIS, Metab and MetaBox using the same mass spectral library (Appendix 3a).

*Metabolite identification*

I evaluated the ability of AMDIS, Metab and MetaBox to identify metabolites analyzed by GC-MS by calculating the percentage of false negatives and false positives produced by each algorithm in each analyzed sample (n = 6). The *percentage of false negatives* is the percentage of metabolites in the standard samples (Table 3.3) that were not identified by an algorithm, while the *percentage of false positives* is the percentage of metabolites misidentified in relation to the total number of compounds identified by the algorithm. For example, a standard sample contains 16 metabolites. An algorithm reports 200 metabolites, which include 14 from the standard sample and an additional 186 misidentified metabolites. In this case, there is 12.5% false negatives (2/16) and 93% (186/200) false positives. A high
percentage of both false negatives and false positives may considerably mislead the final interpretation of metabolomics results. Therefore, an ideal tool for analyzing metabolomics data should report no false negatives nor false positives.

The *match factor* parameter may considerably affect the number of false negatives and false positives. Therefore, I compared the results produced by AMDIS, Metab and MetaBox using the *match factor* values of 60, 70, 80 and 90.

*Metabolite quantification*

Every sample generated as described in section 3.3.1 was analyzed by AMDIS, Metab and MetaBox using *match factors* of 90, 90 and 70, respectively. These match factors were chosen because they resulted in the lowest percentage of false positives produced by each algorithm. As a control, the intensity of all metabolites was manually obtained using the software MSD ChemStation® (Agilent Technologies). All the results were normalized by the intensity of the internal standard 2,3,3,3-d4 alanine (Smart et al. 2010), which was added in equal concentrations to each sample (20 µl) prior to derivatization. In order to ascertain statistical differences between the relative abundances calculated by all algorithms, ANOVAs and a Tukey’s HSD tests were conducted. In addition, the percentage of similarity reported by each of these algorithms was calculated in relation to the control, where the maximum similarity value is 100%. The percentage of similarity was calculated by dividing the lowest relative abundances by the highest relative abundances. For example, if the relative abundance of metabolite A is reported as 0.6 by AMDIS and 0.8 in the control, the percentage of similarity of compound A will be 0.6 divided by 0.8 times 100. On the other hand, if the relative abundance of metabolite A is reported as 0.9 by AMDIS and 0.5 in the control, the percentage of similarity of compound A will be 0.5 divided by 0.9 times 100. Samples originated from each standard mixture (*Mix* 1 and *Mix* 2) were analyzed separately and the average of similarity (*n = 3*) for each metabolite was used to compare the performance of each algorithm. Only metabolites identified by all algorithms were considered for quantification analysis. In the next sections, relative abundances will be referred as abundances in order to facilitate the description of results and interpretation.

### 3.3.2 Results

*Compound identification*
Two different analyses were performed in order to compare the efficiencies of AMDIS, Metab and MetaBox in identifying metabolites. For this, I calculated the percentage of false negatives and positives produced by each algorithm when using match factors 60, 70, 80 and 90.

When applying match factors 60 and 70, AMDIS reported an average (n = 6) of 6.25% ± 0 of false negatives, while match factors 80 and 90 resulted in 10.42% ± 3.27 and 19.79% ± 4.70, respectively (±s.d.) (Figure 3.4). Metab reported similar results, where the lowest percentage of false negatives was reported for match factor 60 (7.29% ± 2.55), and the highest percentage was reported for match factor 90 (19.79% ± 4.70). For MetaBox, the match factor 70 resulted in the lowest percentage of false negatives (19.79% ± 2.55), and the match factor 90 produced the highest percentage (44.79% ± 7.30).

Figure 3.4. False negatives reported by different tools when configured to use different match factors. Two standard mixtures containing 16 compounds in different concentrations were divided in triplicates and analyzed by GC-MS. Each sample was then processed by AMDIS, Metab and MetaBox using different match factors. The average percentages of false negatives produced by each tool using each match factor is presented here. Error bars represent standard deviation. False negatives indicates the percentage of compounds in the standard mixtures that an algorithm was not able to identify. It was calculated by dividing the number of metabolites that an algorithm could not identify by the total number of metabolites in the standard mixtures, in this case 16.

The second analysis shows that AMDIS reported an average (n = 6) of 91.55% ± 0.25 of false positives when applying match factor 60 (Figure 3.5). The lowest percentage was reported using match factor 90, which resulted in 73.78% ± 3.18. Metab reported 79.61% ± 0.79 of false positives when using match factor 60, and match factor 90 resulted in the lowest percentage of false positives (72.09% ± 1.59). MetaBox reported lower percentages of false positives when using match factor 90.
positives for every match factor tested. Using match factor 60, it reported 56.42% ± 2.30, while using match factor 90 this percentage dropped to 42.47% ± 7.94.

**Figure 3.5. False positives reported by different tools when configured to use different match factors.** Two standard mixtures containing 16 compounds in different concentrations were divided in triplicates and analyzed by GC-MS. Each sample was then processed by AMDIS, Metab and MetaBox using different match factors. The average percentages of false positives produced by each algorithm using each match factor are presented here. Error bars represent standard deviation. False positives indicates the number of compounds misidentified by an algorithm. It was calculated by dividing the number of misidentified compounds (compounds not present in the standard mixtures) by the total number of compounds reported by the algorithm.

**Compound quantification**

As revealed by an ANOVA and Tukey’s HSD analyses, the quantification of one metabolite from Mix 1 (L-Norvaline) and one metabolite from Mix 2 (Alanine) was significantly different (p<0.05) between MetaBox and control, and Metab and control (Tables 3.4 and 3.5). Most metabolites (91%), showed no significant difference in the quantifications reported by AMDIS, Metab and MetaBox in relation to the control.
Table 3.4. ANOVA and Tukey’s HSD test results for metabolites in samples originated from Mix 1. The difference between means resulted from pair wise comparisons (Tukey’s HSD test). * p-value < 0.05; ** p-value < 0.01; *** p-value < 0.001.

<table>
<thead>
<tr>
<th>Metabolites in Mix 1</th>
<th>p(ANOVA)</th>
<th>AMDIS-Control</th>
<th>Metab-Control</th>
<th>MetaBox-Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-Norvaline</td>
<td>0.005 **</td>
<td>0.022</td>
<td>-0.001</td>
<td>-0.347 *</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.258, 0.302)</td>
<td>(-0.281, 0.279)</td>
<td>(-0.627, -0.067)</td>
</tr>
<tr>
<td>2-Oxovaleric acid</td>
<td>0.453</td>
<td>0.002</td>
<td>-0.001</td>
<td>-0.001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.043, 0.048)</td>
<td>(-0.047, 0.044)</td>
<td>(-0.047, 0.044)</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>0.512</td>
<td>0.001</td>
<td>0.009</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.267, 0.269)</td>
<td>(-0.259, 0.277)</td>
<td>(-0.260, 0.276)</td>
</tr>
<tr>
<td>Alanine</td>
<td>0.053</td>
<td>-0.029</td>
<td>-0.43</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.251, 0.193)</td>
<td>(-0.652, -0.208)</td>
<td>(-0.222, 0.222)</td>
</tr>
<tr>
<td>ß-Alanine</td>
<td>0.034 *</td>
<td>-0.017</td>
<td>-0.003</td>
<td>0.013</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.140, 0.106)</td>
<td>(-0.126, 0.120)</td>
<td>(-0.116, 0.142)</td>
</tr>
<tr>
<td>Citric acid</td>
<td>0.403</td>
<td>-0.018</td>
<td>-0.02</td>
<td>0.003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.503, 0.468)</td>
<td>(-0.506, 0.465)</td>
<td>(-0.482, 0.489)</td>
</tr>
<tr>
<td>2,3,3,3-d4 alanine</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>0.154</td>
<td>0.006</td>
<td>-0.002</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.014, 0.026)</td>
<td>(-0.022, 0.018)</td>
<td>(-0.019, 0.021)</td>
</tr>
<tr>
<td>Succinic acid</td>
<td>0.238</td>
<td>0.017</td>
<td>0.001</td>
<td>-0.002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.008, 0.042)</td>
<td>(-0.024, 0.026)</td>
<td>(-0.027, 0.024)</td>
</tr>
<tr>
<td>Threonine</td>
<td>0.189</td>
<td>-0.019</td>
<td>-0.011</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.089, 0.051)</td>
<td>(-0.081, 0.059)</td>
<td>(-0.080, 0.060)</td>
</tr>
</tbody>
</table>
Table 3.5. ANOVA and Tukey’s HSD test results for metabolites in samples originated from Mix 2. The difference between means resulted from pair wise comparisons (Tukey’s HSD test). * p-value < 0.05; ** p-value < 0.01; *** p-value < 0.001.

<table>
<thead>
<tr>
<th>Metabolites in Mix 2</th>
<th>p(ANOVA)</th>
<th>AMDIS-Control</th>
<th>Metab-Control</th>
<th>MetaBox-Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-Norvaline</td>
<td>0.42</td>
<td>0.022</td>
<td>-0.001</td>
<td>-0.347</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.258, 0.302)</td>
<td>(-0.281, 0.280)</td>
<td>(-0.627, -0.067)</td>
</tr>
<tr>
<td>2-Oxovaleric acid</td>
<td>0.016*</td>
<td>0.002</td>
<td>-0.001</td>
<td>-0.002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.043, 0.048)</td>
<td>(-0.047, 0.044)</td>
<td>(-0.047, 0.044)</td>
</tr>
<tr>
<td>3-Hydroxyoctanoic acid</td>
<td>0.214</td>
<td>-0.102</td>
<td>-0.14</td>
<td>-0.504</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.897, 0.693)</td>
<td>(-0.935, 0.654)</td>
<td>(-1.066, 0.058)</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>0.104</td>
<td>0.001</td>
<td>0.009</td>
<td>0.007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.267, 0.269)</td>
<td>(-0.259, 0.277)</td>
<td>(-0.260, 0.276)</td>
</tr>
<tr>
<td>Alanine</td>
<td>0.043*</td>
<td>-0.029</td>
<td>-0.43 ***</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.251, 0.193)</td>
<td>(-0.652, -0.208)</td>
<td>(-0.222, 0.222)</td>
</tr>
<tr>
<td>Citric acid</td>
<td>0.04*</td>
<td>-0.018</td>
<td>-0.02</td>
<td>0.004</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(-0.503, 0.468)</td>
<td>(-0.506, 0.465)</td>
<td>(-0.482, 0.489)</td>
</tr>
<tr>
<td>2,3,3,3-d4 alanine</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Metabolite</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
</tr>
<tr>
<td>----------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>0.015 *</td>
<td>0.006</td>
<td>-0.002</td>
<td>0.001</td>
</tr>
<tr>
<td></td>
<td>(-0.014, 0.026)</td>
<td>(-0.022, 0.018)</td>
<td>(-0.019, 0.021)</td>
<td></td>
</tr>
<tr>
<td>Succinic acid</td>
<td>0.002 **</td>
<td>0.017</td>
<td>0.001</td>
<td>-0.002</td>
</tr>
<tr>
<td></td>
<td>(-0.008, 0.042)</td>
<td>(-0.024, 0.026)</td>
<td>(-0.027, 0.024)</td>
<td></td>
</tr>
<tr>
<td>Threonine</td>
<td>0.05</td>
<td>-0.019</td>
<td>-0.011</td>
<td>-0.01</td>
</tr>
<tr>
<td></td>
<td>(-0.089, 0.051)</td>
<td>(-0.081, 0.059)</td>
<td>(-0.080, 0.060)</td>
<td></td>
</tr>
<tr>
<td>Tryptophan</td>
<td>0.285</td>
<td>-0.065</td>
<td>-0.025</td>
<td>-0.022</td>
</tr>
<tr>
<td></td>
<td>(-0.181, 0.051)</td>
<td>(-0.140, 0.091)</td>
<td>(-0.138, 0.093)</td>
<td></td>
</tr>
<tr>
<td>Vanillic acid</td>
<td>0.079</td>
<td>-0.012</td>
<td>-0.002</td>
<td>0.002</td>
</tr>
<tr>
<td></td>
<td>(-0.114, 0.090)</td>
<td>(-0.104, 0.101)</td>
<td>(-0.101, 0.104)</td>
<td></td>
</tr>
</tbody>
</table>

Figures 3.6 and 3.7 show the similarity percentage calculated for metabolites in Mix 1 and Mix 2, respectively. One hundred percent of similarity indicates that the abundance reported by a specific algorithm was exactly the same as in the control. L-Norvaline shows low percentage of similarity in both Mix 1 and Mix 2 when quantified by MetaBox. The same occurred with Alanine when quantified by Metab.
Figure 3.6. Quantification of metabolites in Mix 1. Metabolites present in samples originated from the standard mixture Mix 1 were quantified by AMDIS, Metab and MetaBox using match factors of 90, 90 and 60, respectively. The results produced by each algorithm were then compared against manually quantified metabolites (control). The bars represent the average of similarity calculated for each metabolite and the error bars represent their respective standard deviation.

Figure 3.7. Quantification of metabolites in Mix 2. Metabolites present in samples originated from the standard mixture Mix 2 were quantified by AMDIS, Metab and MetaBox using match factors of 90, 90 and 60, respectively. The results produced by each algorithm were then compared against manually quantified metabolites (control). The bars represent the average of similarity calculated for each metabolite and the error bars represent their respective standard deviation.
3.3.3 Discussion

Compound identification

AMDIS and Metab yielded less than 20% of false negatives at every match factor tested (Figure 3.4). For both algorithms, the match factor 60 yielded the lowest percentage of false negatives. Compared to AMDIS and Metab, MetaBox produced higher percentages of false negatives, where the match factor 60 also promoted the lowest percentage (19.79% ± 2.55). The percentage of false negatives depends on the match factor used and, for MetaBox, the correlation threshold used. The use of lower correlation thresholds and lower match factors would certainly reduce the percentage of false negatives reported by AMDIS, Metab and MetaBox.

Despite producing lower percentages of false negatives, AMDIS and Metab yielded more than 70% of false positives for all match factors tested. Although Metab uses AMDIS for compound identification, it outperformed AMDIS by yielding lower numbers of false positives across all match factor, particularly when using match factors of 60 and 70, where it reported 10% fewer false positives (Figure 3.5). MetaBox outperformed both AMDIS and Metab, yielding at least 20% fewer false positives than both AMDIS and Metab for every match factor used (Figure 3.5).

The match factor parameter used by AMDIS, Metab and MetaBox, and the correlation threshold used by MetaBox, are used to define the accuracy applied in identification of metabolites. The higher the match factor and correlation threshold, the higher the similarity required between spectral library and biological sample to positively identify metabolites. Consequently, a higher match factor and correlation threshold results in lower percentages of false positives; however, it increases the percentage of false negatives, as reported by AMDIS, Metab and MetaBox (Figures 3.4 and 3.5).

The lower percentage of false negatives and higher percentage of false positives reported by AMDIS may reflect the reason behind its development. The algorithm used by AMDIS was originally developed to identify chemical weapons in complex chemical mixtures (Stein, 1999). In this case, single samples are analyzed at a time in order to quickly identify any sign of potential target compounds or chemical weapons. Then, additional analyses can be further performed in order to confirm the identification of target metabolites. In this case, a low percentage of false negatives is crucial, and AMDIS performs considerably well on this.
However, most metabolomics experiments aim to identify the highest possible number of metabolites present in biological samples and compare their abundances or relative abundances across experimental conditions. It is ideally a non-target analysis generally limited only by the metabolites represented in the spectral library. The biological interpretation is then achieved based on the metabolite profile generated by each sample. In this case, the percentages of both false negatives and false positives are crucial for a reliable biological interpretation. A high percentage of false negatives represents potential losses of biological evidence, while a high percentage of false positives may considerably mislead the interpretation. For example, from 100 metabolites reported by AMDIS, 70 are false metabolites, not actually present in the analyzed samples. Therefore, results generated by AMDIS should be manually curated and critically assessed in order to achieve sound biological interpretations. Metab was able to keep up with the low percentage of false negatives generated by AMDIS and reduced the percentage of false positives by about 10% for match factors 60, 70 and 80, which indicates great improvement. Although MetaBox showed 15% more false negatives than AMDIS and Metab, it produced at least 20% less false positives for all the match factors tested, a sensible improvement that may considerably impact metabolomics studies. Additionally, the correlation threshold used by MetaBox can be reduced in order to decrease the percentages of false negatives produced by this algorithm.

The degree to which MetaBox is able to deconvolute overlapping or superimposed spectra depends on the reproducibility of the RT generated by the GC-MS equipment and on the optimization of the mass spectral library in use. As MetaBox relies strongly on the expected RT of each metabolite, a reproducible RT generated by the GC-MS considerably reduces the impact of overlapping spectra. In addition, the mass spectral library in use must be inspected and optimized so that metabolites expected at similar RTs are represented by different IMFs. Under these conditions, the method used by MetaBox can be highly efficient in deconvoluting and identifying metabolites.

Compound quantification

The quantifications performed by AMDIS, Metab and MetaBox were compared against the control. In total, the abundances of 22 metabolites were evaluated via ANOVAs and Tukey’s HSD tests. The difference between means reported by Tukey’s HSD test is an indication of quantification accuracy. Results closest to 0 are considered more similar to the control and,
hence, more accurate. AMDIS showed the most accurate quantification for only three metabolites (Tables 3.4 and 3.5), which represents 13.63% of the metabolites evaluated. Metab showed the most accurate quantification for six metabolites (27.27%), while MetaBox showed the most accurate quantification for ten metabolites (45.45%). Metab and MetaBox equally quantified three metabolites and their abundances were the most accurate when compared to AMDIS. In total, Metab and MetaBox performed the most accurate quantification for 40.90% and 59.09% of the metabolites evaluated, respectively. The higher accuracy of Metab and MetaBox in relation to AMDIS is well represented in figures 3.7 and 3.8, where the great majority of the metabolites show a higher percentage of similarity when analyzed by MetaBox and then Metab.

The lowest quantification accuracy reported by AMDIS may be the result of its strategy for quantifying metabolites. AMDIS makes use of different IMFs as reference when quantifying the same metabolite across samples, which results in higher variability and higher difference between quantification and control. The use of different IMFs for metabolite quantification is another indication that the algorithm used by AMDIS was not originally developed for metabolomics studies. AMDIS does not seem to be developed for comparing the abundances of metabolites across different samples, which is one of the most used approaches in metabolomics. The fact that AMDIS was not originally made to analyze metabolomics data also becomes apparent in the format used for reporting results. AMDIS generates an individual report for each analyzed sample. Consequently, when applied for metabolomics, AMDIS’ results must be manually combined in a single table or spreadsheet, and this can be enormously time-consuming depending on the number of samples being processed. On the other hand, Metab and MetaBox were developed specially for metabolomics studies. Their results are reported in a single spreadsheet containing the metabolites identified and their respective abundances in every analyzed sample, which is the most common format used for further data normalization and analysis.

In summary, the results reported in this chapter demonstrate that the new algorithm, MetaBox, produces considerably more accurate results than both AMDIS and Metab for processing metabolomics data. It greatly reduces the percentage of false positives and considerably improves the quantification of metabolites analyzed by GC-MS.

The processing of data is considered one of the most critical and time-consuming steps in metabolome analysis as it directly impacts the biological interpretation of metabolomics.
studies. Manual data curation (e.g. manual identification, quantification and normalization) prior to statistical data analysis may require more than six months, depending on the size of the data set. Although AMDIS has been the most common freeware used to process metabolomics data analyzed by GC-MS (Smart et al. 2010), the results presented here show that its performance no longer matches the requirements of metabolomics experiments: reliable identification of metabolites, accurate quantification of metabolites and the report of results in a format that facilitates further data analysis. A replacement for AMDIS is urgently needed, and while there remains room for improvement, MetaBox is evidently a very promising candidate.

3.4. MetaBox – The package

The algorithm presented in this chapter, MetaBox, is considerably time consuming if performed manually. Therefore, I implemented this algorithm in an R package called MetaBox (Figure 3.8). It contains 11 functions that deconvolute and identify metabolites, normalize metabolomics data and perform statistical analysis. Here I describe how to install MetaBox and how to use each of its functions.

3.4.1 Requirements

MetaBox is available for MacOS (R version 2.15.1) and depends on XCMS (Smith et al., 2006), tcltk (R Development Core Team, 2012), R.utils (R Development Core Team, 2012), Splines (R Development Core Team), plyr (Wickham, 2011), simecol (Petzoldt and Rinke, 2007), FactoMineR (Husson et al., 2012) and TeachingDemos (Snow, 2012). Every function of MetaBox is available as Appendix 3.

3.4.2 Description

In this section, I describe the usage of the 11 functions in MetaBox. Any argument with default value of “popup” generates a popup dialog box enabling the user to click-and-point to select the desired directories, open files, select compounds for normalizations or to select options for required arguments. The user can, however, elect to enter a character or numeric string containing the information required by each argument. The arguments save, folder and output have the same behavior in every function where they appear. When save =
TRUE, the results produced by the function are saved to a CSV file in the directory specified by `folder` and named as specified by `output`.

- `buildLib(amdisLib = "popup", folder = "popup", save = TRUE, output = "ion_lib")`

AMDIS is one of the most used software for deconvolution, identification and quantification of metabolites analyzed by GC-MS. It allows users to easily build spectral libraries from standards.
Figure 3.8. MetaBox functions. MetaBox contains 11 functions that deconvolute and identify metabolites in biological samples analyzed by GC-MS, normalize metabolomics data and perform statistical analysis.

Every AMDIS library is composed of two files: a file with the extension .CID and another with the extension .MSL. The function buildLib converts any AMDIS library into a CSV
file exactly in the format required by MetaBox. For that, buildLib (Appendix 3b) requires only the .MSL file from the AMDIS library, which can be specified by the argument amdisLib. As a result, buildLib generates a data frame containing for each compound: the name of the compound, its expected retention time, the four most abundant mass fragments and the expected ratios between them. An example of the data frame generated by buildLib is available as vector ion.lib. The user can type data(ion.lib) at the command prompt; it will load an example of data frame generated by buildLib into the current R session. Then, typing ion.lib at the command prompt the actual data frame is presented in the command window.

- identify(main.folder = "popup", ion.lib = "popup", excludeLib = "", align = TRUE, internal.standard = "popup", matchFactor = 0.5, correlation = 0.95, scoreCut = 11, filterNoise = TRUE, save = TRUE, output = "MetaBox_data")

identify (Appendix 3c) is used to deconvolute and identify compounds in samples analyzed by GC-MS. A GC-MS analysis generates one data file per sample. To avoid the user having to define which file corresponds to which experimental condition, identify requires that files, in AIA format (.CDF), corresponding to the same condition are stored within their own folder and this is (usually) named after the condition. Thus, there should be one folder per condition. All such folders must be stored within another (main) folder whose path is defined by a character string supplied to the main.folder argument. identify loads the GC-MS files into an R session and uses the library specified by ion.lib to deconvolute and identify compounds. The ion.lib must have precisely the same format as the library produced by the function buildLib. The argument excludeLib can be used to exclude from the analysis compounds that are certainly not present in analyzed samples, such as compounds derivatized by a different derivatization method. When align = TRUE, the retention time of the chromatogram is shifted in relation to the internal standard defined in internal.standard. For example, if align = TRUE, internal.standard = "D4-Alanine MCF derivative" and the retention time of D4-Alanine MCF derivative is of one minute more than its expected retention time, the whole chromatogram has its retention time increased by 1 minute.
The argument `filterNoise` is used to filter the background noise potentially present in analyzed samples. The background noise may be originated, for example, by the natural degradation of the column used in the GC-MS, which may constantly release some compounds. These compounds end up reaching the mass detector and generate a signal that might interfere with the identification of compounds actually present in the sample. A better identification can be performed if this background noise is removed prior to compound identification. When `filterNoise = TRUE`, `identify` detects the fragments present in the background noise with their respective intensities. Then, every time these fragments are found in the chromatogram they have their intensities subtracted by their respective intensity detected in the background noise.

Three main steps confirm the identification of a specific compound. First, the four most abundant fragments corresponding to a specific compound are analyzed. These fragments must be present in the sample at the same retention time. Second, these fragments must have between them the same intensity ratio as defined in the `ion.lib`. For example, 2,3,3,3-d4alanine is identified by the fragments 106 m/z, 62 m/z, 74 m/z and 107 m/z. These fragments must be present in the sample in the following ratios: (int62m/z)/(int106m/z) = 0.163; (int74m/z)/(int106m/z) = 0.114; and (int107m/z)/(int106m/z) = 0.041; where `int106m/z` is the intensity of the fragment 106 m/z, `int62m/z` is the intensity of the fragment 62 m/z, `int74m/z` is the intensity of the fragment 74 m/z, and `int107` is the intensity of the fragment 107m/z. However, the ratio between the most abundant fragments is not always precise as each fragment may have the interference of other compounds showing similar fragmentation. Thus, the argument `matchFactor` is used to define how similar this ratio must be to consider a compound as real. A value of 0 assumes that no ratio is used to identify a compound, while 0.99 assumes that the ratio between the intensity of the most abundant fragments must be precisely the same as the ones specified in the `ion.lib`. Third, the fragments corresponding to a compound must have a positive correlation between them. During a specific time window, the intensity of these fragments must vary in the same trend. Thus, the argument `correlation` can be used to define how strong this correlation must be to positively identify a compound. It accepts values from 0 to 0.99, where 0 requires no correlation and 0.99 requires a precise correlation between fragments. Each one of these confirmation steps performed by `identify` generates points that are associated to each potential compound. The argument `scoreCut` is then used to define the minimum score that a compound must receive to be considered real. It accepts values from 9 to 12, where higher
scores mean less false positives. As a result, identify generates a data frame containing all the identified metabolites and their respective abundances in each analyzed sample.

- `extractIon(main.folder = "popup", compound = "popup", ion.lib = ", timeWindow = ", save = TRUE, output = "fragment_intensity")`

The function `extractIon` (Appendix 3d) is used to support compound identification and quantification. It extracts from .CDF files the intensity of a specific fragment during a specified retention time window. The argument main.folder behaves as described for `identify`. In order to facilitate the location of the desired compound and its correspondent fragments, the user can simply specify the name of the desired compound through the argument compound. The expected retention time and the most abundant fragment of this compound are then automatically collected from the mass spectral library specified through ion.lib. Alternatively, a specific retention time window can be specified through the argument timeWindow, which must receive a numeric group such as `c(10, 11)` for 10 to 11 minutes. If the argument compound is left as default, a dialog box will pop up prompting the user to select the mass fragments to be analyzed. As a result, `extractIon` plots the intensity of each fragment and displays the highest intensity of the most abundant fragment or the intensity of the first fragment specified by the user in each analyzed sample.

All the following functions accept as input data a data frame such as the one generated by `identify`, where the first column contains the name of the compounds identified, and the following columns contain the abundance of each compound in the different samples analyzed.

- `raw.peaks(main.folder = "popup", correct.RT = TRUE, method = "loess", save = TRUE, output = "mass_fragments")`

`raw.peaks` (Appendix 3e) produces a data frame containing all the mass fragments detected by the GC-MS and their intensities in each biological sample. It requires the same folder structure as `identify`. The arguments main.folder and save also behave as per `identify`. When the argument correct.RT = TRUE (default) the retention time of all samples is corrected using the method specified by method (see the function `rector` in the
XCMS package for more details). Further analyses can then be performed using any of the above functions.

- `del.false(data = "popup", true = 0.65, medium.tag = "none", true.medium = 0.68, save = FALSE, folder = "popup", output = "no_false")`

`del.false` (Appendix 3f) is used to exclude compounds that are detected in less than a user-specified proportion of samples since such compounds are considered to be false positives. The argument `true` takes a value from 0.0 to 1.0 indicating the proportion of samples in which each compound must be identified in order for it to be considered a true compound. For example, consider an experiment with six replicates per condition and `true` = 0.50. For each condition, compounds detected in fewer than 3 replicates will have their intensities replaced by NA.

The `medium.tag` argument is required when studying extracellular metabolites (e.g. footprinting) and is used to specify the columns of the input data that correspond to samples from the uncultured medium. Due to the high reproducibility of compound profiles and intensities of samples from uncultured media, often fewer replicates are needed for these than are required for the experimental conditions. Therefore, a different proportion of false positives to `true` may be appropriate and can be specified through `true.medium`.

`del.false` generates a data frame containing only those metabolites present in at least the proportion of samples specified by `true` and `true.medium`.

- `norm.internal(data = "popup", internal.std = "popup", save = FALSE, folder = "popup", output = "norm_int")`

`norm.internal` (Appendix 3g) generates a data frame containing metabolite intensities normalized by a user-nominated internal standard. The `internal.std` argument is used to specify this metabolite. If `internal.std = "popup"` (default) a list of metabolites will be presented to the user.

- `norm.medium(data = "popup", medium = "popup", log.transform = TRUE, save = FALSE, folder = "popup", output = "norm_medium")`
norm.medium (Appendix 3h) is applied when performing footprint analysis. It subtracts the average log-intensity of each metabolite identified in the uncultured medium from the log-intensity of the same metabolite identified in each biological sample. medium takes a character string indicating which columns of the input data frame correspond to the uncultured medium. The data are log-transformed (default) because intensities often differ by at least one order of magnitude across conditions. Compounds having negative log-intensities after normalization by internal standard indicate cell consumption, while positive intensities indicate cell secretion, of metabolites.

- **norm.biomass(data = "popup", biomass = "popup", save = FALSE, folder = "popup", output = "norm_bio")**

Biological samples may have different biomass content due to the different experimental conditions and/or technical variability. Thus, norm.biomass (Appendix 3i) produces a data frame where the intensities of the compounds within a sample are scaled by that sample’s biomass. biomass defines the biomass of each biological sample.

- **htest(data = "popup", signif.level = 0.05, log.transform = TRUE, save = FALSE, folder = "popup", output = "htest")**

When more than two conditions are under investigation ANOVA is used to test differential metabolite intensities between conditions, otherwise a t-test is used. A column of p-values resulting from these is added to the data frame specified through data. htest (Appendix 3j) generates a data frame comprising only compounds statistically significantly different at the specified signif.level. If log.transform = TRUE (default), the t-test or ANOVA is calculated using the log-transformed input data. Care should be taken since, if htest is applied to the data frame generated by norm.medium, the input data might already be on the log-scale. If so, set log.transform = FALSE.

- **MetaBoxPCA(data = "popup", no.zeros = TRUE, plot.pcs = c(1,2))**

The function MetaBoxPCA (Appendix 3k) is used to apply principal component analysis (PCA) on a data frame generated by any function of MetaBox. When no.zeros = TRUE, only compounds identified in every condition are analyzed. The argument plot.pcs is used to define the PCS to be plot. For example, plot.pcs = c(1,2) indicates that the
PC 1 and PC 2 are to be plot, while \texttt{plot.pcs = c(2,3)} indicates that PC 2 and PC 3 are to be plot. The user can save the plots by using the save option of the R console.

### 3.5. Conclusion

In this chapter, I presented a new algorithm, MetaBox, able to deconvolute and identify metabolites analyzed by GC-MS. I validated MetaBox by comparing its performance against two commonly used tools, AMDIS and Metab. Remarkably, MetaBox showed greater accuracy in both identification and quantification of metabolites. However, MetaBox is considerably time consuming if performed manually. Therefore, I implemented it in an R package called MetaBox, which automates the algorithm and incorporates functions I had previously developed for the R package Metab. As an R package, MetaBox allows users to construct flexible pipelines for data analysis and allows pop up dialog boxes, which considerably facilitates its usage by R beginners. Ultimately, MetaBox produces more reliable results, enhances support for biological interpretations and saves time. In other words, this new tool supports higher quality research with fewer resources.
Chapter 4. Pathway Activity Profiling (PAPi) - from the metabolite profile to the metabolic pathway activity

Content of this chapter is part of: Aggio et al., 2010. Pathway Activity Profiling (PAPi): from the metabolite profile to the metabolic pathway activity. Bioinformatics, 26(23), 2969-2976. doi: 10.1093/bioinformatics/btq567
4.1. Introduction

Metabolomics is one of the newest omics technologies and has been evolving rapidly during recent years. Combined with robust analytical methods, metabolomics is capable of screening large numbers of low molecular mass metabolites (≤ 1.5 KDa) (Nielsen, 2007) in biological samples. Metabolites are intermediates of biochemical reactions and are essential in linking different pathways within a biological system. However, metabolites are synthesized and modified by enzymes, which are products of gene transcription. Accordingly, many regulatory processes involving metabolites, enzymes, mRNAs and genes play an important part determining the levels of metabolite. For this reason, metabolomics has been considered essential for the validation of data sets generated by other omics technologies (Çakir et al., 2006) and has been largely applied as a functional genomics tool and as part of systems biology studies (Oliver et al., 1998; Nielsen & Oliver, 2005; Andersen & Nielsen, 2009).

However, the convoluted nature of cell metabolism, where one metabolite can participate in many different pathways, makes the pathway activity analysis, in particular, the most difficult ‘omics-data’ to interpret (Villas-Bôas et al., 2005). In addition, as with any other post-genomics technology, metabolomics generates large data sets requiring sophisticated bioinformatics tools for their processing and analysis (Kopka et al., 2005).

Despite the analytical aspects of metabolomics (e.g. quenching of metabolism, metabolite extraction and data acquisition) being well advanced (Dunn, 2008), the correlation between metabolite level and metabolic pathway activity is still considered a complex task to achieve. Consequently, the biological interpretation of metabolomics data remains a major bottleneck in metabolome analysis (Çakir et al., 2006). As a result, to assist with post-genomic data analysis of cell metabolism, a great number of proprietary and open source software packages have been developed during the last 10 years by different companies (e.g. AnalyzerPro® for GC-MS and LC-MS data mining by SpectralWorks Ltd; MarkerView™ for metabolomics and protein/peptide biomarker profiling by Applied Biosystems; Mass Profiler Professional by Agilent Technology) and institutions (e.g. Bioconductor). In addition, many web-based databases are now available and provide important information regarding metabolite diversity, metabolic pathways, biochemical reactions, enzymes and genes (Kopka et al., 2005). Among these, the Kyoto Encyclopedia of Gene and Genomes (KEGG) (Ogata et al., 1999) is one of the most popular databases and it is freely available through http://www.genome.jp/kegg/. KEGG has application programming interfaces (API) that
allow its use by external software. Consequently, several computational tools have been created to automatically access, extract and manipulate the information contained in these databases (http://www.genome.jp/kegg/soap/; Arita, 2004). R (R Development Core Team, 2012), an open-source software environment developed for statistical computing (www.r-project.org), is among those with hundreds of available packages developed for different purposes, in particular “KEGGSOAP” (Zhang and Gentleman, 2009) and “KEGG.db” (Carlson et al., 2009), which enable access to and use of data from the KEGG database in a flexible way.

These significant advances in bioinformatics tools have improved the quality of both the data generated by omics studies and the subsequent biological interpretations. However, when relating metabolite level to pathway activity, there are only few tools available (e.g. MetPA, Pathway Hunter Tool and Ingenuity Pathway Analysis) and most of them require extensive data pre-processing and demand great knowledge about cellular metabolisms, which increases the time-spent and decreases the accessibility to the biological interpretation.

In this chapter I present a new algorithm, Pathway Activity Profiling (PAPi), which aims to assist the interpretation of metabolomics data based solely on metabolite profiles. PAPi predicts and compares the activity of metabolic pathways between different experimental conditions using metabolite profiles and KEGG database. For this, I define a new measure for pathway activity, which I call Activity Score (AS). Calculated for each pathway, the AS is based on the number of metabolites identified from each pathway and their relative abundances. As a result, the AS represents the likelihood that a metabolic pathway is active inside the cell and, consequently, allows the comparison of metabolic pathway activities.

However, PAPi algorithm is considered time consuming if performed manually. Therefore, I implement it in an R package, PAPi, which considerably facilitates its usage. PAPi uses the data extracted from metabolomics experiments together with the KEGG database to generate relative ASs. PAPi includes functions that perform either a t-test or analysis of variance (ANOVA) on the ASs, and generate graphical summaries of the results. The functions also enable the use of optional pop-up dialog boxes making them more accessible to new R users.

PAPi has been published in Aggio et al. (2010) and successfully applied to identify changes in yeast metabolism via exposure to different sound frequencies (Aggio et al., 2012), in bacterial metabolism in response to environmental stress (unpublished data), and was able to
identify the metabolic pathway driving the morphology of *Candida albicans* to filamentous form (Han et al., 2012). Institutions from more than 15 different countries have used PAPi.

4.2. PAPi – The algorithm

4.2.1 Input data

The starting point of PAPi is a table (Table 4.1) containing the KEGG code of the identified metabolites in the first column and their abundances/relative abundances in each sample in the subsequent columns. The KEGG compound code can be found at the KEGG website (http://www.genome.jp/dbget-bin/www_bfind?compound). I assume that the data have been normalized (e.g. normalization by internal standard or uncultured medium) and appropriately transformed (e.g. log transformation) before being submitted to PAPi.

Table 4.1. Input data. The input data used by PAPi is a table containing the KEGG code of metabolites in the first column and their respective abundances/relative abundances in each sample in the subsequent columns.

<table>
<thead>
<tr>
<th>Name</th>
<th>Sample 1.1</th>
<th>Sample 1.2</th>
<th>Sample 1.3</th>
<th>Sample 2.1</th>
<th>Sample 2.2</th>
<th>Sample 2.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C00116</td>
<td>0.43</td>
<td>0.54</td>
<td>0.45</td>
<td>1.31</td>
<td>1.72</td>
<td>0.83</td>
</tr>
<tr>
<td>C00180</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.03</td>
<td>0.10</td>
<td>0.02</td>
</tr>
<tr>
<td>C00041</td>
<td>0.52</td>
<td>0.27</td>
<td>0.57</td>
<td>1.15</td>
<td>1.22</td>
<td>1.42</td>
</tr>
<tr>
<td>C00037</td>
<td>0.05</td>
<td>0.07</td>
<td>0.07</td>
<td>0.29</td>
<td>0.26</td>
<td>0.28</td>
</tr>
<tr>
<td>C02261</td>
<td>0.07</td>
<td>0.06</td>
<td>0.06</td>
<td>0.05</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>C01571</td>
<td>0.12</td>
<td>0.01</td>
<td>0.01</td>
<td>0.05</td>
<td>0.03</td>
<td>0.05</td>
</tr>
<tr>
<td>C00183</td>
<td>1.36</td>
<td>1.67</td>
<td>1.81</td>
<td>7.58</td>
<td>6.90</td>
<td>5.23</td>
</tr>
<tr>
<td>C00123</td>
<td>1.05</td>
<td>1.35</td>
<td>1.45</td>
<td>2.48</td>
<td>2.55</td>
<td>1.77</td>
</tr>
<tr>
<td>C00407</td>
<td>2.95</td>
<td>3.48</td>
<td>3.96</td>
<td>2.41</td>
<td>2.14</td>
<td>2.55</td>
</tr>
</tbody>
</table>
4.2.2 Description

To facilitate the algorithm’s description, I have divided it into six steps shown in Figure 4.1.

**Figure 4.1. Description of PAPi algorithm.** Starting from a metabolomics data set (initial data frame), (A) PAPi searches the KEGG database for potential active metabolic pathways related to each metabolite. Then, (B) the abundance of each metabolite is assigned to associated pathways. (C) The total number of metabolites associated with each pathway is recorded and the pathways are then ranked according to the number of metabolites with which they are associated. (D) The Activity Score of each pathway is then calculated and (E) normalized by its respective percentage of metabolites calculated in (C). This process is repeated for each sample analyzed and (F) the results are combined in a unique framework. Finally, statistical analyses (G) can be performed using the framework produced in (F).

In the first step, the KEGG database is accessed and the pathway(s) associated with each metabolite is (are) returned. That is, all pathways for which each metabolite is known to play a part are collected from the KEGG database (Figure 4.1A). Each identified pathway then receives a score based on the abundance/relative abundance of the metabolite to which it is linked (Figure 4.1B). The total number of metabolites associated with each pathway is recorded and the pathways are then ranked according to the number of metabolites with which they are associated. The percentage of detected metabolic intermediates is then calculated for each listed pathway (Figure 4.1C). Finally, I sum over the scores for each pathway to obtain the total pathway score, SA, (Figure 4.1D) and normalize it by dividing by the proportion of metabolites detected from its respective pathway (Figure 4.1E). The normalized score of each pathway represents the level of its activity inside the cell, where the
higher the score the lower the activity. Thus, I define the normalized activity score for pathway P as

\[ S_A(P) = \left( r_1 + r_2 + \ldots + r_N \right) N / k, \]

where

- \( r_i = \) the relative abundance of metabolite \( i \) detected from pathway \( P \),
- \( N = \) the number of metabolites detected in pathway \( P \), and
- \( k = \) the total number of metabolites known to play a part in pathway \( P \).

The six operations described above are applied for all samples from each condition studied. Afterwards, the outcome is combined in a final data frame containing the list of all active pathways and their respective normalized scores for each sample. When applied to the analysis of extracellular metabolites (metabolic footprinting), the profile of metabolites should be normalized (subtracted) by the uncultured medium (control sample) before analysis by PAPi. In my example, a two-sample t-test was used to assess pathway differential activity between two conditions, and only those that were statistically significant were retained (Figure 4.1G). Note that when three or more conditions are being investigated ANOVAs can be applied to test the global hypothesis of a difference between conditions, and then pairwise comparisons of conditions can be performed for those pathways for which a statistical difference in activity between conditions was declared. Finally, a graph showing all-metabolic pathways and their respective normalized scores in both conditions can be generated to represent the data and assist the interpretation.

However, if I simply plot the pathways’ activity scores, the resultant graph would be somewhat counterintuitive, considering the activity score is inversely related to the predicted pathway activity, as higher the activity score the lower is the predicted pathway activity. Thus, before plotting, I suggest to scale and “invert” the pathway activity scores. Scaling is performed by setting the activity score of one of the conditions to 0, referring to this as the reference condition, and then scaling the activity score of the other condition relative to the reference, referring to the latter as the target condition. Afterward, I “invert” each pathway’s activity score before plotting, i.e.
and

\[ S'_{A, \text{Reference}}(P) = 0 \]

For example, if the activity scores of the reference and target in pathway P are \( S_{A, \text{Reference}}(P) = 3 \) and \( S_{A, \text{Target}} (P) = 6 \), respectively, their scaled scores will be 0 (Reference) and -2 (Target).

While scaling and inverting of pathway activity scores is not essential, I recommend their use because they facilitate a more intuitive visualization of the data.

**4.2.3 Results**

A metabolite profile produced by metabolomics contains two main types of information that can be used to infer the activity of metabolic pathways in a specific experimental condition: the number of metabolites detected from each metabolic pathway and the levels of these metabolites. My methodology was developed to infer the difference in metabolic pathway activity between distinct experimental conditions when using single-time-point metabolite profiles. It is based on two main assumptions. I postulate that (i) if a given metabolic pathway is more active in a given condition, a larger number of metabolic intermediates from that pathway is likely to be detected by metabolomics. Higher activity of a metabolic pathway is generally associated to the expression of a higher number of genes related to this pathway, which is expected to result in a higher number of intermediates to be detected by metabolomics techniques. For example, Dressaire and co-workers (2011) demonstrated that more genes associated to isoleucine production in *L. lactis* are expressed during isoleucine starvation. This increase in gene expression translates into a higher number of reactions (and their intermediates) towards isoleucine production. However, since the same metabolites are usually detected across different conditions, I assume that (ii) the greater the activity of a metabolic pathway, the lower the abundance of the metabolic intermediates from that pathway. The higher levels of enzymes associated to higher gene expression in a specific condition is expected to result in higher conversion rates of metabolic intermediates inside the cells, which may result in lower abundances of metabolites detected by metabolomics techniques. For example, Neves and co-workers (2002) studied the effects of oxygen in the
metabolism of \textit{L. lactis} when no heme is added to the medium. They reported a higher flux through glycolysis anaerobically accompanied by lower levels of key intermediates such as 3-phosphoglycerate and phosphoenolpyruvate. However, considering that a metabolic pathway presenting lower flux might result in the accumulation of specific intermediates (due to lower levels of enzymes and consequently lower intracellular conversion rates), the predicted activity of that pathway can be misled by the higher abundances of a few metabolites. Therefore, to minimize this effect, I normalize the pathway activity scores by the percentage (%) of metabolites detected from each pathway, because it is assumed (first assumption) that I will detect a higher proportion of metabolic intermediates from a highly active pathway than from pathways of low activity. However, most metabolomics data are based on relative quantification and each metabolite is subject to different response factors depending on the analytical techniques being used. Therefore, unless absolute quantification data is available, PAPi results obtained from relative quantification based metabolomics data can only be used to compare metabolic pathway activity between different data classes. In other words, PAPi results can only be used to predict that pathway A is more active in condition/sample class I than in condition/sample class II, but not that pathway A is more active than pathway B. In order to predict that pathway A is more or less active than pathway B, absolute quantification data is required.

Considering the assumptions above, PAPi results in a data frame containing the identified pathways and their respective activity scores for each sample. However, this method usually detects over 100 pathways as being potentially active in a cell based on a data set containing around 50 different identified metabolites and many of these pathways are equally active between different conditions. Thus, statistical analyses are carried out on the activity scores to identify pathways that are differentially active between pairs of conditions. A two-sample t-test is performed when only two conditions are being studied, otherwise ANOVAs can be performed. The final output of the analysis is a data frame containing the pathways most likely to be defining the differences between data classes and this data frame can be plotted in a line graph.

PAPi also generates unique observations when used to analyze metabolic footprinting data (extracellular metabolites from microbial or cell cultures). For this type of data I subtract the abundance of metabolites in the uncultured medium from the abundance of metabolites detected in the spent culture medium before applying PAPi (Smart et al., 2010). This way, some pathways receive a positive activity score while others become negative. A negative
pathway activity score indicates that metabolites playing part in those pathways were more abundant in the uncultured medium than in the spent culture, suggesting that the activity of those pathways is related to the uptake of metabolites from the medium. A positive score, on the other hand, suggests that metabolic intermediates from those pathways were secreted to the extracellular medium during microbial or cell growth, possibly resulting from a metabolic overflow. This way, important information regarding metabolite uptake and intracellular metabolic overflow is generated, enhancing the biological interpretation of metabolic footprinting data.

Although the assumptions used to build PAPi suit most of the metabolic pathways, the glycolysis pathway seems to work in a distinct way. According to Stephanopoulous and co-workers (1998), even when glycolysis is in a high flux state one should expect high abundance of its intermediates, which is rather reasonable if we consider that glycolysis is a central metabolic pathway that provides precursors for many essential pathways (e.g. TCA cycle). Thus, I agree that for glycolysis PAPi may not be accurate, but it is still useful in detecting whether glycolysis is operating at different fluxes between experimental conditions.

In addition, it is important to emphasize that by using the non-species-specific KEGG database information one usually observes that pathways not naturally belonging to the organism under study may appear as potentially active by PAPi. This output sounds wrong in principle, but I speculate that it can actually provide important information about possible metabolic interactions between different organisms or species and also about novel metabolic reactions. For instance, extracellular metabolites produced by an organism A can simultaneously play a role in the metabolism of an organism B, which can be a potential metabolic link that allows the interaction between these two organisms. In addition, when a pathway appears as being active it does not mean that the whole pathway is active, but rather specific reactions of that pathway are taking place.

PAPi analysis is based only on metabolites identified by metabolomics techniques. However, one could predict missing metabolites by the presence of their precursors or products in metabolic pathways reported by PAPi. If these are metabolites in considerably low concentration, one may go back to the GC-MS data and manually verify the chromatogram at the expected RT for the presence of these compounds.

Metabolic pathway activity is directly related to metabolic flux distribution. Thus, pathways presenting lower scores based on intracellular metabolomics data are likely to be operating at
high metabolic flux. Thereby, this method not only reduces the time spent on metabolomics data analysis but it may also enable one to compare the metabolic flux of different pathways in different conditions (indirect fluxomics). More importantly, PAPi is a method developed to support the interpretation of metabolomics data, one of the major bottlenecks in metabolomics. PAPi has been successfully applied for detecting metabolic changes in *Candida albicans* during yeast-to-hypha transition (Han et al. 2012). Papi’s activity scores revealed a global downregulation of the primary metabolism, which was further confirmed by the analysis of the ATP pool. In addition PAPi has been applied for detecting metabolic changes in *Saccharomyces cerevisiae* (Aggio et al. 2010). PAPi’s activity scores successfully reported a lower activity of metabolic pathways derived from glutamate when comparing a wild-type strain against a mutant strain with a deleted NADPH-dependent glutamate dehydrogenase and an over expressed NADH-dependent glutamate dehydrogenase. PAPi’s activity scores were also useful to generate one of the main hypotheses presented in Chapter 5, the role of lactic acid in the excretion of proline by *L. lactis*. Finally, the PAPi package presented in this chapter has already been downloaded and used by universities and institutes from more than 20 different countries.

### 4.3. PAPi - The package

PAPi package comprises three functions: papi, papi.htest and papi.line (see Appendixes 4). papi is used to calculate the ASs for each sample and stores these, together with the list of all active pathways, in a data frame. papi.htest performs either a t-test or ANOVA on the ASs and can be used to identify pathways that are differentially active, since these are generally thought to be important in terms of observed differences in metabolism between different conditions. Finally, the results can be summarized using papi.line, which generates a line graph of average AS plotted against pathway. Pathway activity profiles of all experimental conditions can be superimposed on the same graph (Figure 4.2).

PAPi can be applied to the analysis of intra- (i.e. metabolic fingerprinting) and extracellular metabolites (i.e. metabolic footprinting).
4.3.1 Requirements

PAPi was developed under R version 2.10.1 and depends on seven other packages, namely KEGGSOAP (Zhang & Gentleman, 2009) and KEGG.db (Carlson et al., 2009) from Bioconductor, and reshape (Wickham, 2007), gdata (Warnes, 2012), gplots (Warnes, 2009), plotrix (Lemon, 2010), FactoMineR (Husson et al., 2012) and tcltk (R Foundation for...
4.3.2 Description

In this section I describe the usage of the four functions in \texttt{PAPi: papi, papi.htest and papi.line}.

\begin{verbatim}
papi(data = "import", save = TRUE, folder = "popup", output = "papi_results")
\end{verbatim}

\texttt{papi} (see Appendix 4a) is applied to a data frame in which the first column contains the identified metabolites' KEGG codes and all subsequent columns contain their abundances in each analyzed sample. (The KEGG codes can be found at \url{http://www.genome.jp/dbget-bin/www_bfind?compound} and consist of the letter “C” followed by a sequence of 5 digits, e.g. Glucose = C00031.) GC-MS analysis generates one data file per sample. To avoid the user having to define which file corresponds to which experimental condition, the first row of the input data must contain the value “Replicates” in the first column and the name of the experimental condition to which each sample belongs.

\texttt{papi} comprises four arguments: \texttt{data}, \texttt{save}, \texttt{folder} and \texttt{output}. The \texttt{data = "import"} (default) argument results in a pop up dialog box, allowing the user to click-and-point to the comma separated value (CSV) format file from which the data is to be read. Alternatively, \texttt{data} can take the name of a data frame containing the samples' metabolite abundances. Similarly, the default behavior of \texttt{folder} is for a pop up dialog box to be presented to the user. The user can then select the directory to which the results will be saved, if \texttt{save = TRUE}. Alternatively, \texttt{folder} takes a character string naming the path to the directory where the results are to be saved. \texttt{output} take a character string naming the file produced by \texttt{papi}.

\texttt{papi} generates a data frame containing the pathways identified across all samples, their KEGG codes and their \texttt{Ass}. 

Statistical Computing, 2012). All of these packages can be installed from the Bioconductor web site (\url{http://www.bioconductor.org/}).
papi.h.test(data = "import", signif.level = 0.05, save = FALSE, log.transform = TRUE, folder = "popup", output = "htest")

The data, save, folder and output arguments behave as described in the papi function. As described for papi, papi.h.test (see Appendix 4b) uses the first row of the input data to identify the number of experimental conditions. For two experimental conditions, papi.h.test then performs a t-test to test for differential pathway activity between conditions, otherwise ANOVA is used. A column of p-values resulting from the analyses carried out on the ASs from each pathway is added to the initial input data frame.

The level of significance, specified by the signif.level argument, is used to create a data frame consisting of only the differentially active pathways. If log.transform = TRUE, the statistical test is applied on the log transformed data. When the argument save = TRUE (default) this data frame is saved to a CSV file named according to output.

papi.line(data = "import", relative = TRUE, folder = "popup", output = "papi_line_graph", legend.position=c(20,20), cex.legend=0.8, position.ylab=c(-3,5), margins=c(14,2.5,2,2), colour=c("black", "gray"), cex.xlab=0.6)

The data, save and out.folder arguments behave as described in the papi.h.test. papi.line (see Appendix 4c) generates a line graph in which, for each condition, the average normalized total ASs are plotted against the identified pathways (Figure 4.2). The relative = TRUE (default) argument results in a line graph of pathway activity profiles for each target condition relative to the selected reference condition. By default, the first element of the conditions argument is defined as the reference condition. A horizontal line is drawn at zero to serve as a visual reference. Plots of the untransformed ASs can be generated setting relative = FALSE. The line graph produced by papi.line is saved to a file called papi_line_graph.png (default) in the directory specified by folder. The argument legend.position defines the position of the legend in the graph by indicating its place in relation to the y axis and x axis (e.g. legend.position = c(10,2)). The size of the legend is defined through the argument cex.legend. The position of the y axis lab is defined using the argument position.ylab, which behaves as described for legend.position. The argument colour defines the colour that will represent each
condition (e.g. colour=c("black", "gray")). Finally, the argument cex.xlab defines the size of the x axis label, which shows the metabolic pathways detected.

4.4. Conclusion

In this chapter, I introduced PAPi, a new algorithm able to correlate metabolomics datasets and metabolic pathways activities in a high-throughput way. Doing so, PAPi is able to compare the activity of metabolic pathways under different conditions, which provides great support for hypothesis generation and facilitates the biological interpretation of metabolomics data. Furthermore, as KEGG database also supports enquires using protein and gene transcription levels, PAPi has the potential to combine data from different omics in one unique framework, further simplifying the biological interpretation of the data. Modifications to include PAPi as part of the Bioconductor database (http://www.bioconductor.org/) are currently in progress.
Chapter 5. Metabolite profiles of *Lactococcus lactis* growing aerobically and anaerobically in different carbon sources.
5.1. Introduction

*Lactococcus lactis* is a gram-positive lactic acid bacterium (LAB) able to quickly convert sugar (e.g. galactose, glucose, lactose and cellobiose) into lactic acid. This bacterium converts over 90% of the carbon source mainly into lactic acid and only about 5% into biomass (Novak and Loubiere, 2000). This key characteristic of *L. lactis* has been largely exploited by the dairy industry to produce fermented food products such as fermented milk and cheese (Kowalczyk et al., 2008). This highly efficient production of lactate quickly acidifies the raw food material, curdling the milk and inhibiting growth of unwanted microorganisms, consequently preserving and extending the shelf life of food products (Oliveira et al., 2005).

*L. lactis* is traditionally classified as a facultative anaerobe (Neves et al., 2005). In the absence of oxygen, *L. lactis* uses sugars to produce mostly lactic acid (Duwat et al., 2001), while oxygenation shifts the sugar utilization towards heterolactic fermentation, resulting in the additional production of by-products such as acetic acid, formic acid, CO₂, ethanol, acetoin (Smart and Thomas, 1987). In the presence of oxygen, this bacterium is able to undergo respiration provided that a *heme* source - a prosthetic group that consists of a porphyrin containing an iron ion in the center (Caughey et al., 1975) - is present in the culture (Duwat et al., 2001). It is equipped with genes involved in respiratory growth, but lacks genes involved in heme biosynthesis (Duwat et al., 2001).

Understanding the production of by-products in *L. lactis’* fermentation has become essential for the current dairy industry, as these metabolites define the organoleptic properties of the final fermented product. A great number of the studies involving fermentation of *L. lactis* target the influences of oxygen (Smart and Thomas, 1987; Neves et al., 2002a), sources of carbohydrate (Kowalczyk et al., 2008) and key glycolytic intermediates (Neves et al., 2002b) on the production of these specific by-products. For example, in 2005 Oliveira and co-workers developed the first genome-scale metabolic model of *L. lactis*, which was built mainly to predict the production of these by-products.

Genome-scale mathematical models are based on literature-derived data. Therefore, this current focus on metabolic pathways involved directly on the biosynthesis of by-products constraints the expansion of existing genome-scale models. For example, there is a considerable lack of knowledge regarding fatty acids synthesized by *L. lactis*. There is a clear need for studies applying more global approaches, such as metabolomics, to identify a
broader range of metabolites involved in its metabolism. To the best of my knowledge, there is only one single study applying metabolomics using gas chromatography – mass spectrometry (GC-MS) to L. lactis (Azizan et al., 2012). This study assessed changes in L. lactis metabolism promoted by different incubation temperatures and culture agitation.

In this chapter I report the results of a metabolomics experiment conducted on L. lactis in the presence and absence of oxygen growing on three different carbon sources: glucose, galactose and lactose + cellobiose. The aim of this work was to produce metabolomics-generated data to expand the existing genome-scale metabolic network of L. lactis. Hence, I used the availability of oxygen and different carbon sources to 1) generate metabolite profiles L. lactis and 2) generate novel biological hypotheses regarding the metabolism of this organism. The production of by-products has been extensively studied on L. lactis growing on glucose (Nordkvist et al., 2003), galactose (Neves et al., 2010) and lactose (Aleksandrzak-Piekarczyk et al., 2005) and, therefore, these carbon sources were chosen to allow further comparisons.

5.2. Materials and methods

5.2.1 The organism

Lactococcus lactis subsp. lactis IL1403 was kindly provided by AgResearch Grasslands as a frozen culture. Bacteria were maintained on Lactobacilli MRS (Difco™, Becton Dickinson and company, Le Pont de Claix, France) agar plates at 30°C and used to prepare all pre-inoculums.

5.2.2 Chemically defined medium

L. lactis was cultivated using shake-flasks containing a chemically defined medium (CDM) adapted from a previously described medium (Raya et al., 1998) and contained (per liter) trisodium citrate, 0.26g; ammonium acetate, 0.94g; citric acid, 0.37g; monopotassium phosphate, 0.90g; dipotassium phosphate, 7.50g; magnesium chloride, 0.20g; iron chloride, 5mg; calcium chloride, 50mg; cobalt chloride, 2mg; alanine, 0.24g; arginine, 0.12g; asparagine, 0.34g; cysteine, 0.17g; glutamine, 0.51g; glycine, 0.17g; histidine, 0.11g; isoleucine, 0.20g; leucine, 0.47g; lysine, 0.35g; methionine, 0.12g; phenylalanine, 0.28g; proline, 0.68g; serine, 0.34g; threonine, 0.23g; tryptophan, 1.0g; tyrosine, 0.29g; valine, 0.33g; para-aminobenzoic acid, 10mg; biotin, 10mg; folic acid, 1mg; nicotinic acid, 1mg;
panthothenic acid, 1mg; riboflavin, 1mg; thiamine, 1mg; pyridoxine, 2mg; cyanocobalamin, 1mg; orotic acid, 5mg; 2-deoxy thymine, 5mg; inosine, 5mg; DL-6,8-thioctic acid, 2.50mg; pyridoxamine, 5mg; adenine, 10mg; guanine, 10mg; uracil, 10mg; and xanthine, 10mg. The final pH was adjusted to 6.3.

5.2.3 Growth conditions

The bacteria were grown (n = 5) on CMD under two different conditions, aerobically and anaerobically, supplemented with three different carbon sources individually: 1% galactose; 1% glucose; and 1% lactose plus 1% cellobiose. Cellobiose was used in addition to lactose because this strain of L. lactis is able to uptake lactose only in the presence of cellobiose (Kowalczyk et al., 2008).

The culture flasks were inoculated with an overnight-grown pre-inoculum prepared in CDM supplemented with 1% of glucose and pH of 6.3. Cultivations were performed using the same incubator at 150 rpm and 30°C, in 500 mL shake flasks containing 200 mL of medium and cotton plugs (aerobic cultures) or air stoppers (anaerobic cultures). Anaerobic conditions were achieved by blowing nitrogen gas into the shake flasks for 10 min before inserting the air stoppers. Each condition was carried out in a different day but using vitamins and amino acids prepared in a single bulk to ensure identical background medium composition, and inoculated with identical initial optical density at 600nm (OD$_{600}$) using a pre-inoculum grown overnight in the same medium.

5.2.4 Fermentation parameters

The pH and the OD of culture broth were measured using a Toledo FiveEasy™ (model LE409) pH meter and a Hitachi® (model U-1100) spectrophotometer at 600nm, respectively, each hour until the stationary growth phase was reached. The presence of contamination was monitored by gram-staining using light microscopy, and the bacterial growth rate ($\mu$) was calculated as

$$\mu = \frac{2.303(\log(OD) - \log(OD_0))}{(t - t_0)}$$

where $OD$ is the optical density measured at the end of the exponential growth, $OD_0$ is the optical density measured at the beginning of the exponential growth, $t$ is the time at the end of
the exponential growth and $t_0$ is the time at the beginning of the exponential growth (Stanbury and Whitaker, 1984).

5.2.5 Sampling for metabolite profiling

*Intracellular metabolites*

Two 50 mL culture broth samples were rapid-harvested from each flask when the cultures had reached an OD$_{600}$ of 0.4 to 0.5 (middle to late exponential growth phase). The cells metabolism was quenched using cold-glycerol saline solution (-23°C) followed by cold centrifugation at -20°C. Internal standard, 20 µl of 2,3,3,3-d4 alanine (10 mM), was added to the cell pellets, and the intracellular metabolites were extracted using freeze/thaw cycles and cold methanol-water (1:1, v/v) solution (Smart et al., 2010).

*Extracellular metabolites*

Initially, three samples of 1mL each were harvested from the uncultured medium. Then, concurrent with the intracellular samples, three 1mL samples were rapid harvested from each flask and filtered using a 0.20µm syringe filter.

5.2.6 Biomass quantification and sample concentration

The biomass content of each sample was determined after intracellular metabolite extraction by measuring the dry weight of cell debris as described in Smart et al. (2010). The cell extracts containing intracellular metabolites and the extracellular samples were freeze-dried using a 12L Labconco Freeze Dryer (Labconco Corporation).

5.2.7 Chemical derivatization and analysis of metabolites

The freeze-dried solids were resuspended in 200 µl of 1M sodium hydroxide and derivatized using methyl chloroformate (Smart et al., 2010). The derivatized samples were then analyzed by the GC-MS systems: Agilent GC7890 coupled to an MSD5975 (Agilent technologies), with a quadrupole mass selective detector (EI) operated at 70 eV and a column ZB-1701 (Phenomenex), 30 m x 250 m (internal diameter) x 0.15 (film thickness), with 5 m guard column; The MS was operated in scan mode (start after 6 min; mass range 38-650 a.m.u. at 1.47 scans/s) (Smart et al., 2010).
5.2.8 Data analysis

The entire pipeline for data analysis was performed using MetaBox (Chapter 3) and PAPi (Chapter 4). As an additional measure for control and for proper biological interpretation, I manually confirmed the results generated by MetaBox. For this, I used the R-software package XCMS (Smith et al., 2006) for manually extracting the intensity of the most abundant mass fragments of each identified metabolite.

Mass spectral library

An in-house library was used to identify compounds analyzed by GC-MS (Appendix 3a).

Data normalization

The analytical steps involved in the generation of metabolomics data are potential source of variability (Smart et al., 2010). However, this variability can be minimized by the use of an internal standard (Smart et al., 2010). For that, I divided the intensity of each compound by the intensity of 2,3,3,3-d4 alanine detected in each sample. In addition, in order to compare the abundance of intracellular metabolites extracted from cells growing on different conditions, one must normalize it by the number of cells or the dry biomass of each sample (Smart et al., 2010). Accordingly, I divided the intensity of each compound by the biomass content of its respective sample. As the CDM used here is a complex medium, one must normalize extracellular samples by the uncultured medium in order to identify compounds being excreted or consumed by the cells (Smart et al., 2010). For that, I subtract the intensity of each metabolite identified in the uncultured medium from their respective intensity when identified in extracellular samples.

Statistical analysis and metabolic pathway profiling

In order to discover metabolites showing different abundances/relative abundances in response to the availability of oxygen, the comparisons described in Figure 5.1 were performed using Student’s t-test. The metabolite profiles generated for each experimental condition was further analyzed using the R package PAPi (Chapter 4). Then, the same comparisons shown in Figure 5.1 were performed on the calculated Activity Scores (AS) of metabolic pathways. The AS is calculated based on the number of metabolites identified from each metabolic pathway and their respective abundances. The AS then indicates the activity of a metabolic pathway (Chapter 4 – Section 4.2) under a specific experimental condition. As
relative abundances were used as input data, the activity scores generated by PAPi can only be used to compare the activity of a metabolic pathway across different experimental conditions and not to compare the activity of different metabolic pathways in the same experimental conditions (See section 4.2.3).

Figure 5.1. Data analysis. Student’s t-test was used to identify metabolites and metabolic pathways showing different abundances and activities, respectively, in response to the oxygen availability.

5.3. Results

Growth

The growth rate of cells grown aerobically (Figures 5.2 and 5.3) was significantly higher than the growth rate of cells grown anaerobically in all tested carbon sources (n≥3, Galactose, p-value = 3.06e-03; Glucose, p-value = 6.50e-05; Lactose + Cellobiose, p-value = 5.74e-05) (Table 5.1). On galactose, the growth rate of L. lactis was 90% higher when growing aerobically. The same trend was observed for glucose and lactose + cellobiose, with growth rates of 80% and 69% higher, respectively. In average, aerobic cultures presented a growth rate 80% higher than anaerobic cultures (Figure 5.2 and 5.3).

Although the CDM was strongly buffered, there was a significant difference in the final pH of cultures growing aerobically and anaerobically in all carbon sources tested (Galactose, p-value = 1.13e-03; Glucose, p-value = 5.37e-03; Lactose + Cellobiose, p-value = 4.24e-04; n = 3) (Table 5.1). In average, aerobic cultures presented a final pH 10% higher than anaerobic cultures.
Figure 5.2. Growth curve, growth rate and pH of *Lactococcus lactis* IL1403 growing aerobically. The bacterial growth medium, CDM, was supplemented with 1% glucose, 1% galactose or 1% lactose + 1% cellobiose.

Figure 5.3. Growth curve, growth rate and pH of *Lactococcus lactis* IL1403 growing anaerobically. The bacterial growth medium, CDM, was supplemented with 1% glucose, 1% galactose or 1% lactose + 1% cellobiose.
Table 5.1. Fermentation parameters. Growth rates and standard error of *Lactococcus lactis* and the final pH of aerobic and anaerobic cultures supplemented with galactose, glucose or lactose + cellobiose.

<table>
<thead>
<tr>
<th>Carbohydrate</th>
<th>Growth rate (O.D./hour)</th>
<th>pH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aerobic (n = 3)</td>
<td>Anaerobic (n = 3)</td>
</tr>
<tr>
<td>Galactose</td>
<td>0.389 ± 0.009</td>
<td>0.204 ± 0.005</td>
</tr>
<tr>
<td>Glucose</td>
<td>0.478 ± 0.015</td>
<td>0.266 ± 0.012</td>
</tr>
<tr>
<td>Lactose + Cellobiose</td>
<td>0.339 ± 0.006</td>
<td>0.201 ± 0.019</td>
</tr>
</tbody>
</table>

Metabolites

Considering both intra and extracellular samples, 58 compounds (Table 5.2) were identified by MetaBox (Chapter 3) as being part of the metabolism of *L. lactis* IL1403 grown under aerobic and anaerobic conditions on CDM supplemented with 1% glucose, 1% galactose and 1% lactose + 1% cellobiose (see Appendix 5a for the complete metabolite profile).

Table 5.2. Identified compounds. 58 compounds were identified as being part of the metabolism of *Lactococcus lactis*.

<table>
<thead>
<tr>
<th>Compound name</th>
<th>Compound name</th>
<th>Compound name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Methyloctadecanoic acid</td>
<td>C20:5n-3,6,9,12,15 acid (EPA)</td>
<td>Lactic acid</td>
</tr>
<tr>
<td>3-Hydroxybenzoic acid</td>
<td>C21:0 acid (Heneicosanoic acid)</td>
<td>Leucine</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid</td>
<td>C22:0 acid (Behenic acid)</td>
<td>Lysine</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid</td>
<td>C6:0 acid (Capronic acid)</td>
<td>Malonic acid</td>
</tr>
<tr>
<td>3,5-Diiodo-L-tyrosine</td>
<td>C8:0 acid (Caprylic acid)</td>
<td>Methionine</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>cis-Aconitic acid</td>
<td>N2-Acetyl-L-lysine</td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid</td>
<td>Citraconic acid</td>
<td>Nicotinic acid</td>
</tr>
<tr>
<td>Alanine</td>
<td>Citric acid</td>
<td>p-Toluic acid</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>Cysteine</td>
<td>Phenylalanine</td>
</tr>
<tr>
<td>Benzoic acid</td>
<td>Dodecane</td>
<td>Proline</td>
</tr>
<tr>
<td>fatty acid (Carbon number: Carbon chain)</td>
<td>organic acid</td>
<td>amino acid</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------</td>
<td>------------</td>
</tr>
<tr>
<td>C10:0 acid (Capric acid)</td>
<td>Glutamic acid</td>
<td>Pyroglutamic acid</td>
</tr>
<tr>
<td>C12:0 acid (Lauric acid)</td>
<td>Glutathione</td>
<td>Quinic acid</td>
</tr>
<tr>
<td>C14:0 acid (Myristic acid)</td>
<td>Glycine</td>
<td>Serine</td>
</tr>
<tr>
<td>C16:0 acid (Palmitic acid)</td>
<td>Heptadecane</td>
<td>Succinic acid</td>
</tr>
<tr>
<td>C16:1n-7 acid (Palmitoleic acid)</td>
<td>Histidine</td>
<td>Tartaric acid</td>
</tr>
<tr>
<td>C18:0 acid (Stearic acid)</td>
<td>Isoleucine</td>
<td>Threonine</td>
</tr>
<tr>
<td>C18:1n-9 acid (Oleic acid)</td>
<td>Itaconic acid</td>
<td>Tryptophan</td>
</tr>
<tr>
<td>C18:2n-6,9 acid (Linoleic acid)</td>
<td>L-Norvaline</td>
<td>Tyrosine</td>
</tr>
<tr>
<td>C18:3n-3,6,9 acid (Linolenic acid)</td>
<td>L-Ornithine</td>
<td>Valine</td>
</tr>
<tr>
<td>C20:0 acid (Arachidic acid)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note on results interpretation**

PAPi (Chapter 4) was applied on metabolomics samples previously derivatized by methyl chloroformate (MCF), a reaction that converts non-volatile compounds into volatile ones. The MCF reaction converts amino and non-amino organic acids into volatile esters and carbamates, which allows them to be analyzed by GC-MS (Smart et al., 2010). Therefore, the metabolite profiles analyzed by PAPi contain mainly organic and non-organic acids, missing important compounds, such as sugar phosphates, which are key metabolites in sugar metabolism. This has been taken into consideration during the interpretation of results.

**Comparison a – Galactose aerobic vs. anaerobic**

Figure 5.4 shows metabolites detected at significantly different levels when *L. lactis* was growing aerobically and anaerobically in galactose. In the absence of oxygen, *L. lactis* showed a significantly higher excretion of lactic acid (1.28e3% higher production of lactic acid anaerobically; p-value = 1.4e-03; n = 4), uptake of citric acid and excretion of aspartic acid. Although the CDM contained no fatty acids, there was consumption of fatty acids anaerobically. The fatty acids consumed by the cells may have originated from the impurity of other compounds present in the CDM. On the other hand, the presence of oxygen resulted
in excretion of citric acid, consumption of aspartic acid and higher intracellular accumulation of fatty acids.

Figure 5.5 presents all metabolic pathways showing significantly different ASs (p-value < 0.05; n ≥ 3) when *L. lactis* was growing aerobically and anaerobically on galactose. The intracellular results show a higher activity of most metabolic pathways when *L. lactis* is growing anaerobically. Histidine metabolism was the only metabolic pathway showing lower activity (p-value = 0.005; n ≥ 3) under this condition. The extracellular results show positive ASs for most metabolic pathways in the presence and absence of oxygen, which indicates that these pathways are associated with excretion of metabolites when growing under such conditions. Interestingly, the extracellular profile of pathways shows that histidine metabolism and pathways associated with pyruvate metabolism were more active in excreting metabolites under anaerobic conditions. The higher activity predicted for pyruvate metabolism may be a result of the higher excretion of lactic acid anaerobically.
Figure 5.4. Metabolites showing different relative abundances for *Lactococcus lactis* II.1403 growing aerobically and anaerobically on CDM supplemented with 1% galactose. Intracellular and extracellular compounds detected at significantly (p-value < 0.05; Student’s t-test; n ≥ 3) different levels when comparing aerobically and anaerobically growth on galactose. In the extracellular bar graph, compounds showing positive relative abundances were excreted by the cells, while compounds showing negative relative abundances were consumed by the cells. Error bars represent standard error.
Figure 5.5. Metabolic pathways showing different Activity Scores (AS) for *Lactococcus lactis* IL1403 growing aerobically and anaerobically on CDM supplemented with 1% galactose. Metabolic pathways showing significantly (p-value < 0.05; Student’s t-test; n ≥ 3) different Activity Scores (ASs) when comparing aerobically and anaerobically growth on galactose. The ASs were calculated using the R package PAPi (Chapter 4). In the intracellular graph, the aerobic condition is set as reference. Therefore, the ASs calculated for metabolic pathways when *L. lactis* was growing anaerobically
are shown in reference to their ASs calculated aerobically. In the extracellular graph, metabolic pathways showing positive AS are likely to be related to the excretion of metabolites, while metabolic pathways showing negative ASs are likely to be related to the consumption of metabolites.

**Comparison b - Glucose aerobic vs. anaerobic**

Figure 5.6 shows metabolites detected at significantly different levels (p-value < 0.05; n ≥ 3) when *L. lactis* was growing aerobically and anaerobically in glucose. *L. lactis* excreted almost every compound detected at significantly different levels when growing aerobically (Figure 5.6). Interestingly, quinic acid was the only metabolite consumed at significant (p-value ≤ 0.05; n ≥ 3) levels by *L. lactis* under this condition. Intracellularly, there was a significantly higher accumulation of fatty acids. Anaerobically, most of the metabolites detected at significantly different levels have been consumed by *L. lactis*; the cells excreted only lactic acid, quinic acid and succinic acid under this condition. Quinic acid was not in the composition of the CDM, and therefore, it may have originated from the impurity or degradation of other compounds in the CDM.

Figure 5.7 presents all metabolic pathways showing significantly different ASs (p-value < 0.05; n ≥ 3) when *L. lactis* was growing aerobically and anaerobically in glucose. The intracellular results indicate up-regulation of most metabolic pathways when growing anaerobically. The extracellular results show that most metabolic pathways were associated with the consumption of metabolites. On the other hand, the majority of the pathways were associated with excretion of metabolites when growing aerobically. Pyruvate metabolism was the only metabolic pathway showing the opposite trend, which may be a result of the higher excretion of lactic acid anaerobically.
Figure 5.6. Metabolites showing different relative abundances for *Lactococcus lactis* IL1403 growing aerobically and anaerobically on CDM supplemented with 1% glucose. Intracellular and extracellular compounds detected at significantly (p-value < 0.05; Student’s t-test; n ≥ 3) different levels when comparing aerobically and anaerobically growth on glucose. In the extracellular bar graph, compounds showing positive relative abundances were excreted by the cells, while compounds showing negative relative abundances were consumed by the cells. Error bars represent standard error.
Metabolic pathway showing different Activity Scores (AS) for *Lactococcus lactis* IL1403 growing aerobically and anaerobically on CDM supplemented with 1% glucose. Metabolic pathways showing significantly different ASs when comparing aerobic and anaerobic growth on glucose.

The ASs were calculated using the R package PAPi (Chapter 4). In the intracellular graph, the aerobic condition is set as reference. Therefore, the ASs calculated for metabolic pathways when *L. lactis* was growing anaerobically are shown in reference to their ASs calculated aerobically.

In the extracellular graph, metabolic pathways showing positive ASs are likely related to the excretion of metabolites, while metabolic pathways showing negative ASs are likely related to the consumption of metabolites.

Figure 5.7: Metabolic pathways showing different Activity Scores (AS) for *Lactococcus lactis* IL1403 growing aerobically and anaerobically on CDM supplemented with 1% glucose. Metabolic pathways showing significantly different ASs when comparing aerobic and anaerobic growth on glucose. The ASs were calculated using the R package PAPi (Chapter 4). In the intracellular graph, the aerobic condition is set as reference. Therefore, the ASs calculated for metabolic pathways when *L. lactis* was growing anaerobically are shown in reference to their ASs calculated aerobically. In the extracellular graph, metabolic pathways showing positive ASs are likely related to the excretion of metabolites, while metabolic pathways showing negative ASs are likely related to the consumption of metabolites.
**Comparison of Lactose + cellobiose aerobic vs. anaerobic**

Figure 5.8 shows metabolites detected at significantly different levels (p-value < 0.05; n ≥ 3) when *L. lactis* was growing aerobically and anaerobically on lactose + cellobiose. Aerobically, there was consumption of most compounds from the medium, sparse excretion of amino acids, low accumulation of intracellular metabolites and uptake of citric acid. Anaerobically, there was consumption of citric acid and excretion of amino acids such as proline, ornithine, leucine, isoleucine and cysteine. Under this anoxic condition, there was also higher accumulation of most metabolites intracellularly.

Figure 5.9 presents all metabolic pathways showing significantly different ASs (p-value < 0.05; n ≥ 3) when *L. lactis* was growing aerobically and anaerobically on lactose + cellobiose. The intracellular results show up-regulation of most metabolic pathways during aerobic growth, while the extracellular results show a higher activity of most metabolic pathways towards consumption of metabolites from the medium.
Figure 5.8. Metabolite profile of *Lactococcus lactis* IL1403 growing aerobically and anaerobically on CDM supplemented with 1% lactose + 1% cellobiose. Intracellular and extracellular compounds present in significantly (p-value < 0.05; Student’s t-test; n ≥ 3) different levels when *Lactococcus lactis* are growing aerobically and anaerobically on CDM supplemented with 1% lactose + 1% cellobiose. In the extracellular bar graph, compounds showing positive relative abundances were excreted by the cells, while compounds showing negative relative abundances were consumed by the cells. Error bars represent standard error.
Figure 5.9. Metabolic pathways showing different Activity Scores (AS) for Lactococcus lactis IL1403 growing aerobically and anaerobically on CDM supplemented with 1% lactose plus 1% cellobiose. Metabolic pathways showing significantly (p-value < 0.05; Student’s t-test; n ≥ 3) different Activity Scores (ASs) when comparing aerobically and anaerobically growth on lactose + cellobiose. Metabolic pathways that received a gray dot were detected only anaerobically. The ASs were calculated using the R package PAPi (Chapter 4). In the intracellular graph, the aerobic condition is set as reference. Therefore, the ASs calculated for metabolic pathways when L. lactis was growing anaerobically are showed in reference to their ASs calculated aerobically. In the extracellular graph, metabolic pathways showing positive AS are likely to be related to the excretion of metabolites, while metabolic pathways showing negative ASs are likely to be related to the consumption of metabolites.
5.4. Discussion

Metabolite profiling

Using the metabolomics technique described in section 5.2.5, 58 compounds (Table 5.2) were identified as being part of the metabolism of \textit{L. lactis} IL1403. To the best of my knowledge, this is the first metabolomics experiment that involves the role of oxygen in the metabolism of \textit{L. lactis} IL1403 and that returns such high number of metabolites. Importantly, 16 fatty acids whose synthesis is not yet described for \textit{L. lactis} have been identified. The metabolite profile generated here is now available as resource to expand existing metabolic networks of \textit{L. lactis}.

Potential respiration via cyanocobalamin

\textit{L. lactis} usually undergoes respiration if a heme source is added to the culture (Duwat et al., 2001), in which case it is expected that \textit{L. lactis} show higher growth rate and lower lactic acid production in the presence of oxygen. On the other hand, when a heme source is not available, as in the CDM used here, the bacterial growth and the production of lactic acid are expected to be very similar both in the presence and absence of oxygen (Duwat et al., 2001). Nonetheless, the results presented in section 5.3 show a significantly higher growth rate of \textit{L. lactis} (Table 5.1) and a significantly lower production of lactic acid when growing aerobically in any carbon source tested (Figures 5.4, 5.6 and 5.8). These surprising results indicate that \textit{L. lactis} was potentially undergoing respiration on the presence of oxygen and, as the CDM used here contained no heme, it is likely that the bacteria were using an alternative molecule to synthesize heme and undergo respiration. Although \textit{L. lactis} is not able to synthesize heme de novo, some genes for this pathway are present and functional (Vido et al., 2004). For example, when protoporphyrin IX is provided, \textit{L. lactis} is able to synthesize heme by incorporating iron into this molecule (Duwat et al., 2001). From all the contents present in the CDM I used, cyanocobalamin is the only compound with a chemical structure similar to protoporphyrin IX (Figure 5.10). Cyanocobalamin belongs to the family of tetrapyrroles, which are able to form metal complexes as protoporphyrin IX and heme do (Scott, 1997). Therefore, these results suggest a novel way in which \textit{L. lactis} might be able to produce heme and undergo respiration via cyanocobalamin. Alternatively, cyanocobalamin could directly be undertaking heme’s function in cells’ respiration. This hypothesis certainly requires further experiments to be confirmed.
Figure 5.10. Chemical structures of heme, protoporphyrin IX and cyanocobalamin. The red dashed line indicates the tetrapyrrole structure of each compound.

*Might* *L. lactis* *excrete* *amino* *acids* *for* *maintaining* *sugar* *consumption?*

*L. lactis* is considered a “nutritionally fastidious” organism (Oliveira et al., 2005) because of its limited capacity to synthesize amino acids de novo. However, it seems to be secreting several amino acids to the extracellular medium when grown aerobically on glucose (Figure 5.6). Vido and co-workers reported a similar fact in 2004, where they showed that *L. lactis* secreted proline and other amino acids to the extracellular medium when growing under aerobic conditions. *L. lactis* has a very compartmentalized metabolism, with almost no overlap between the energy metabolism and the biosynthesis of biomass (Hugenholtz and Kleerebezem, 1999). This allows *L. lactis* to maintain high metabolic rates through energetic pathways, however, resulting in low efficiency in energy conservation. Ecologically, *L. lactis* can be considered a niche engineer in that it is able to quickly consume carbon sources with high production of lactic acid, which reduces the pH from the medium and inhibits the growth of potential competitors. However, the tradeoff appears as a low synthesis of biomass (Hugenholtz and Kleerebezem, 1999). When undergoing respiration, *L. lactis* shows high generation of energy at middle to late exponential phase (Duwat et al., 2001), which results in higher production of building blocks, such as amino acids and nucleotides (Lan et al., 2006). The intracellular data (Figure 5.6) shows a higher accumulation of amino acids inside the cells, and excretion of most of these metabolites when *L. lactis* is grown on glucose under aerobic conditions. In addition, there was a significantly higher growth rate of *L. lactis* under
this condition but no difference in the final biomass produced (final O.D. aerobic = 0.62 ± 0.004; final O.D. anaerobic = 0.66 ± 0.035; p-value = 0.2466; t-test; n= 3). These results suggest a potential bottleneck in biosynthetic pathways involved in the incorporation of building blocks into biomass (e.g. gene transcription and protein assembly). The cell’s machinery involved in these pathways may not be able to work at the same pace as the energy metabolism. Consequently, these building blocks accumulate inside the cell and may end up being secreted or leaked to the extracellular medium. Further experiments involving transcriptomics and proteomics may be able to provide more evidence confirming this hypothesis.

It is important to note that the metabolite profiles generated in this chapter (section 5.3) contain mainly organic and non-organic acids. They miss important compounds, e.g. sugars and sugar phosphates, which are key metabolites involved in central metabolic pathways such as glycolysis. This must be considered when interpreting the pathway profiles presented above (section 5.3). For example, intracellular results from aerobic growth in glucose show a lower activity of most metabolic pathways (Figure 5.7). However, the higher growth rate and intracellular accumulation and excretion of most metabolites in this same condition, may be reflecting a higher activity of energetic metabolic pathways. These apparently contradictory results highlight the importance of taking into consideration not only the AS calculated by PAPi, but the class and concentration of analyzed metabolites and, more importantly, the physiological features of the organism under study.

The metabolic network of L. lactis produced in Chapter 2 could potentially be used to validate the hypothesis reported here. However, this metabolic network is not optimized for simulating the metabolism of L. lactis when undergoing respiration. There is a considerable lack of information about the reactions involved in this process. In addition, the results reported here indicate that L. lactis may be undergoing respiration using cyanocobalamine instead of heme, a result never reported before. Consequently, there is no information about energy consumption or energy production under this condition in any existing metabolic networks of L. lactis. In addition, the experiments reported in this chapter did not cover sugar consumption and the metabolomics experiment did not produce absolute quantification of metabolites. A potential approach in this case would be repeating the experiment and comparing aerobic cultures with and without cyanocobalamine, in addition to measuring sugar consumption and the absolute concentration of sugars and amino acids from the central carbon metabolism in each condition.
Quinic acid potentially produced and consumed by L. lactis

The results presented in section 5.3 show consumption and excretion of quinic acid. Quinic acid plays a part in the Phenylalanine-Tyrosine-Tryptophan Biosynthesis, which is a key metabolic pathways related to the biosynthesis of aromatic compounds (Gibson and Pittard, 1968; van Kranenburg et al., 2002). Aromatic compounds are key metabolites in the final organoleptic properties of fermented food products. Thus, further experiments clarifying how quinic acid is imported, metabolized and excreted by L. lactis IL1403 are of great value to the dairy industry. Although the metabolism of quinic acid has been described in other lactic acid bacteria (Carr et al., 1957; Whiting and Coggins, 1973), this is the first time, to the best of my knowledge, that it is described for L. lactis IL1403.

Citric acid might be used as an extra energy source

Citric acid is a compound highly related to the energy metabolism of L. lactis (Pudlik and Lolkema, 2011) and considerably important for the dairy industry (Starrenburg and Hugenholtz, 1991). The metabolism of citric acid is associated with energy generation, production of lipids (e.g. phospholipids for cell’s membranes), regulation of intracellular pH and production of flavor compounds (Pudlik and Lolkema, 2011). The results presented in section 5.3 show that L. lactis consumes citric acid anaerobically and excretes citric acid aerobically when growing in galactose and glucose (Figures 5.4 and 5.6), which is in agreement with the low pH achieved by L. lactis cultures under these conditions (Table 5.1). The citric acid consumed by the cells was potentially used to regulate the intracellular pH and to generate energy, while the citric acid excreted by the cells aerobically may be a result of the high flux through glycolysis and consequent accumulation of pyruvate (Neves et al., 2002b). Interestingly, on lactose + cellobiose there was a higher consumption of citric acid when L. lactis was growing aerobically (Figure 5.8). Lactose + cellobiose was the carbon source that registered the highest pH (6.52 ± 0.058) (Table 5.1). The fact that citric acid was consumed in a culture with high pH, suggests that this compound has a function in the physiology of L. lactis other than regulate the intracellular pH when growing aerobically on lactose + cellobiose. In this case, the consumption of citric acid could be a result of the low energy generation promoted by the slow fermented sugars, lactose + cellobiose, which would
result in co-metabolism of these sugars (Table 5.1). The citric acid was potentially used as an extra source of energy, which promoted the higher growth rate achieved in this condition. Citric acid is converted to acetic acid in a set of reactions involving oxaloacetic acid, pyruvate and NADH (Pudlik and Lolkema, 2011). Considering the importance of citric acid metabolism in the final organoleptic properties of fermented food products, further experiments, such as using carbon labeled citric acid, would certainly help in clarifying this unexpectedly high consumption of citric acid when *L. lactis* is grown aerobically in lactose + cellobiose.

*Accumulation of lactic acid may induce proline excretion*

Growing on lactose + cellobiose, PAPi registered significantly different activity scores of the Arginine and proline metabolic pathway between growths with and without oxygen. The Arginine and proline metabolism catalyzes the synthesis and oxidation of proline. Proline can be synthesized from ornithine and glutamate or it can be oxidized to glutamate to be used as carbon source (Brandriss and Magasanik, 1979). In this case, proline is converted into glutamate by a set of reactions, the first of which entails proline being converted into 1-pyrroline-5-carboxylate in a reaction catalyzed by proline oxidase (EC1.5.99.8). This reaction is greatly inhibited by lactic acid, as demonstrated for *Escherichia coli* and humans (Wood, 1987; Kowaloff et al., 1977). The results reported in section 5.3 show up-regulation of Arginine and proline metabolism (Figure 5.9) and a high consumption of proline when *L. lactis* is growing aerobically on lactose + cellobiose (Figure 5.8). On the other hand, there was excretion of proline and ornithine to the extracellular medium when *L. lactis* was growing anaerobically. Excretion of proline has been reported before (Vido et al., 2004) in *L. lactis*, however, there is not yet an explanation for that. The results described above suggest that the excretion of proline may be a result of the inhibition of proline oxidase by lactic acid. Under anaerobic conditions, there is a high accumulation of lactic acid inside the cells. Lactic acid inhibits proline oxidase, which is likely to result in the accumulation of proline and its subsequent excretion. As ornithine is a precursor for proline, the accumulation of proline results in the accumulation and excretion or leakage of ornithine. The inhibition of proline oxidase then creates a bottleneck in the Arginine and proline metabolism pathway, which results in its down-regulation, as reported by PAPi. Aerobically, a lower accumulation of lactic acid occurs, and thus no inhibition of proline oxidase, so proline is used as an energy source via its conversion to glutamate. Indeed, comparing the aerobic results for all the carbon sources tested here, lactose + cellobiose produced the lowest accumulation of lactic
acid inside the cells and it was the only carbon source that promoted consumption of proline. Anaerobically, glucose promoted the lowest accumulation of lactic acid inside of the cells and was also the only carbon source that promoted consumption of proline. Although described for *E. coli* and *Bacillus subtilis*, there is no report of proline oxidase (EC1.5.99.8) for *L. lactis*. Therefore, I performed a BLAST search and found formate-tetrahydrofolate ligase showing 48% of similarity with proline oxidase and an e-value of 6e-4, which indicates a potentially similar biological function between these two enzymes. Further experiments are surely required to validate this hypothesis.

**Potential accumulation of fatty acids in response to a high flux through glycolysis**

Growing on galactose and glucose, there was a higher intracellular accumulation of fatty acids when *L. lactis* was growing under aerobic conditions (Figure 5.4 and 5.6). This higher production of fatty acids might be a result of the higher growth rate shown by *L. lactis* under these conditions. A higher growth rate requires a higher flux through glycolysis. As pyruvate is the main product of glycolysis and also a precursor for fatty acids (Neves et al., 2005), a higher flux through glycolysis may result in a higher biosynthesis of fatty acids. Interestingly, lactose + cellobiose produced a different trend. Under this condition, the results showed a higher accumulation of fatty acids anaerobically (Figure 5.8). As lactose and cellobiose are slow metabolized sugars (Neves et al., 2005), this result may be a consequence of the low flux through glycolysis both in the presence and absence of oxygen. Further experiments including the measurement of sugar consumption would certainly generate more evidences to improve this hypothesis.

All the hypotheses generated in this chapter are considered by-products of the main aim: to generate a profile of metabolites involved in the metabolism of *L. lactis*, which will be used to expand the existing metabolic network of this bacterium. These hypotheses were formulated based on the relative abundances of metabolites, the growth rate and the biomass production showed by *L. lactis* on the different experimental conditions studied. An addition, which would certainly improve hypothesis generation, would be the normalization of results by sugar consumption.
5.5. Conclusion

In this chapter I presented the first metabolomics experiment of *L. lactis* growing aerobically and anaerobically on three different carbon sources: galactose, glucose and lactose + cellobiose. Fifty-eight metabolites were identified as part of the metabolism of *L. lactis*, including 16 fatty acids. The profile of metabolites generated here is now available to expand the existent genome-scale metabolic network of *L. lactis*. In addition, to the best of my knowledge, it is the first study suggesting that *L. lactis* is able to undergo respiration in the presence of cyanocobalamin and also the first study suggesting a mechanism for the excretion of proline by *L. lactis*. Furthermore, it is the first evidence of consumption and excretion of quinic acid by *L. lactis* and the first results indicating a higher production of fatty acids when *L. lactis* IL1403 is growing aerobically. Although the metabolomics data set generated here has not yet been fully explored, the results and hypotheses presented in this chapter already provide a better understanding of *L. lactis* metabolism and are of great value for the dairy industry, as they may directly affect the final properties of fermented food products.
6.1. Introduction

The availability of whole genome sequences allowed the reconstruction of genome-scale metabolic networks (Price et al., 2004). The analysis of these networks, mostly using constraint-based approaches (e.g. Flux balance analysis), allowed for the analysis of the cell’s metabolism in a systemic manner, which has been successfully applied to drug discovery (Becker and Palsson, 2005), drug production (Borodina et al., 2005; Oddone et al., 2009), food development (Oliveira et al., 2005) and bioremediation (Durot et al., 2009).

Genome-scale metabolic networks can be reconstructed using two main approaches: top-down and bottom-up. The top-down approach uses high-throughput measurements (e.g. transcriptomics and proteomics) and statistical methods to predict relationships between a cell’s molecular components. On the other hand, the bottom-up approach makes use of the annotated genome sequence as a starting point for predicting reactions potentially occurring in the cell’s metabolism. The literature is then used to expand the metabolic network, define reactions’ reversibility and fill potential metabolic gaps (Thiele and Palsson, 2010). Databases such as Kyoto Encyclopedia of Genes and Genomes (KEGG) (Ogata et al., 1999) (http://www.genome.jp/kegg/) and MetaCyc (Krieger et al., 2004) (http://MetaCyc.org/) are largely used in this process. Currently, these two databases are considered among the most complete chemical catalogues available.

Metabolomics has been considered one of the most important omics technologies applied to systems biology studies (Patti et al., 2012). Metabolomics aims to analyze all metabolites (the small molecules involved within the cell’s metabolism), which are the most downstream point in the flow of information starting from gene expression (Fiehn, 2002). Therefore, metabolite profiles are considered a representation of the phenotype of biological systems and are essential for genome-scale cell modeling.

However, the use of metabolomics data for feeding metabolic networks is not straightforward and, to the best of my knowledge, there is not yet a well-established method for that. Manually searching for potential enzymes related to metabolites reported by metabolomics experiments may be considerably time consuming. It demands thorough literature surveys, searches in biochemical databases and the use of different computational tools to find potential interspecific similarities when the data are not available for the desired organism. Therefore, there is a need for methods and computational tools able to assist in the use of metabolomics data for feeding metabolic networks.
In this chapter, I present a new algorithm: Metabolite to Network (M2N). M2N aims to assist in the use of metabolomics data for feeding metabolic networks by: 1) predicting enzymes potentially related to the metabolism of a specific organism, and 2) generating graphical outputs able to assist the inclusion of predicted enzymes into metabolic networks. For this, M2N makes use of KEGG database, the European Bioinformatics Institute (EBI) (http://www.ebi.ac.uk/) and Basic Local Alignment Search Tool (BLAST) (Altschul et al., 1990). EBI is a centre for research and bioinformatics services. It has available a protein database (i.e. Uniprot) containing protein-related data such as amino acid sequences. BLAST is a computational tool developed to find regions of local similarity between sequences. In practice, BLAST is used to assess the degree of similarity between biological sequences (e.g. nucleotide and amino acid sequences). The use of M2N is illustrated in this chapter by analyzing the metabolite profiles produced in Chapter 5 for the bacteria Lactococcus lactis. Since M2N can be time consuming if performed manually, I developed M2N Toolbox, a MATLAB (MATLAB 7.10.0; R2010a) toolbox that implements this new algorithm and facilitates its usage.

6.2. M2N – The algorithm

6.2.1 Input data

The starting point of M2N is a table (data frame) containing the names of known metabolites (i.e. metabolite profile reported by metabolomics) in the first column and their respective KEGG codes in the second column (Table 6.1). The KEGG compound code of each metabolite can be found at the KEGG website (http://www.genome.jp/dbget-bin/www_bfind?compound). M2N checks whether predicted enzymes have been assigned to the organism under study. For this reason, M2N also requires the KEGG code for the organism (e.g. ‘lla’ for Lactococcus lactis IL1403), which can be found at the KEGG website http://www.genome.jp/kegg/catalog/org_list.html.
Table 6.1. Input data. The input data used by M2N consists in an array of data containing the names of metabolites in the first column and their respective identification code from the KEGG database in the second column.

<table>
<thead>
<tr>
<th>Name</th>
<th>kegg</th>
</tr>
</thead>
<tbody>
<tr>
<td>C16:0 acid (Palmitic acid)</td>
<td>C00249</td>
</tr>
<tr>
<td>C18:0 acid (Stearic acid)</td>
<td>C01530</td>
</tr>
<tr>
<td>C8:0 acid (Caprylic acid)</td>
<td>C06423</td>
</tr>
<tr>
<td>cis-Aconitic acid</td>
<td>C00417</td>
</tr>
<tr>
<td>Citraconic acid</td>
<td>C02226</td>
</tr>
<tr>
<td>Citric acid</td>
<td>C00158</td>
</tr>
<tr>
<td>Cysteine</td>
<td>C00097</td>
</tr>
<tr>
<td>Glutamic acid</td>
<td>C00025</td>
</tr>
<tr>
<td>Glycine</td>
<td>C00037</td>
</tr>
</tbody>
</table>

6.2.2 Description

To simplify the algorithm’s description, I have divided it into 7 steps shown in Figure 6.1. In the first step (Figure 6.1 – Step 1), KEGG database is accessed and all the Enzyme Commission numbers (ECs) associated with each compound from the input data are collected. The EC describes the type of reaction catalyzed by an enzyme and is a common enzyme identifier used in biochemical databases (Bairoch, 2000). KEGG database is then re-accessed to confirm which previously collected ECs are associated with the specified organism (Figure 6.1 – Step 2). KEGG database contains the gene locus id associated with each enzyme previously described for the organism. For example, the EC 4.2.1.3 is associated with the locus id lla:L68478 of Lactococcus lactis IL1403. Therefore, during this second step, every EC previously collected and described for the organism under study receives its respective gene locus. On the third and fourth steps (Figure 6.1 –Steps 3 and 4), the protein sequences associated with each EC are collected from the EBI database and used to BLAST against predefined databases (e.g. nr and Swiss-Prot) and organisms (e.g. L. lactis). ECs showing positive association with the chosen organism and/or associated with protein sequences that show any significantly similarity-hit in the BLAST results are
shortlisted and stored in a file named shortlist.csv. The Expected value (e-value) is used to determine the level of similarity between amino acid sequences (see http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=FAQ#expect for more details about e-value). The e-value calculates the probability of retrieving from the database an amino acid sequence that has a bit-score equal to or greater than the one calculated from the alignment of the query and subject sequences. Subject sequences that are very similar to the query sequence will be associated to very small e-values. The use of the e-value instead of the bit-score is recommended for two reasons: to avoid the influence that the size of the database searched may have in the results, where larger databases are more likely to return positive matches due to chance; and to use a value that is independent of the parameters used for Blasting, such as gap penalty and different substitution matrices. Drastic changes in the size of the database used may affect the e-value associated to a BLAST search. The file shortlist.csv contains the ECs associated with each compound, their respective locus id in KEGG database, their amino acid sequences and the results of the BLAST comparisons. An additional file called full_report.csv stores all the ECs collected from KEGG database, the gene locus associating each of these ECs to the organism under study, the amino acid sequences of each EC and the results of every BLAST comparison performed by M2N (Figure 6.1 – Step 4). In the fifth step (Figure 6.1 – Step 5), KEGG database is accessed and the KEGG code of every metabolic pathway associated with each compound and enzyme contained in the shortlist.csv file are collected. These metabolic pathways are subsequently ranked according to the highest number of metabolites and enzymes present in the shortlist. In the next step, the map of each metabolic pathway is collected from KEGG database, and the compounds and enzymes also present in the shortlist.csv are highlighted using different colors (Figure 6.1 – Step 6). Finally, M2N generates a graphical output of a metabolic network connecting predicted enzymes and metabolites present in the input data (Figure 6.1 – Step 7).
Figure 6.1. Illustrated description of M2N algorithm. M2N uses a typical metabolic profile to predict EC numbers and reactions potentially related to the organism under study. See section 6.2.2 for details.
6.2.3 M2N algorithm’s output

A metabolite profile represents a list of metabolites that were consumed or secreted by the cells, or formed inside the cells as a result of a sequence of chemical reactions mostly catalyzed by enzymes. Barring external sources of contamination, metabolites reported by metabolomics experiments are part of the cell’s metabolism and may have enzymes involving their formation or consumption. Therefore, metabolite profiles can be used to feed metabolic networks by finding these associated enzymes.

The algorithm presented here has three main procedures to predict enzymes related to the metabolite profile of the organism under study: 1) it uses KEGG database as data source to find all potential enzymes related to each metabolite in the metabolite profile; 2) it uses KEGG database to verify which of these enzymes have been described for this specific organism; and 3) it uses BLAST for searching proteins already reported as being associated with the organism under study and potentially showing the same biological function of enzymes collected from KEGG database. BLAST searches are based on the assumption that one can infer the similarity of protein function from the similarity of amino acid sequences (Thiele and Palsson, 2010).

As a result, M2N generates a list of enzymes (represented by EC numbers) potentially involved in the metabolism of an organism and related to the metabolites contained in the input data. In addition, M2N generates metabolic maps demonstrating the relationship between predicted enzymes and metabolites in pre-existing metabolic pathways (e.g. glycolysis, TCA cycle, etc.), which assists in finding metabolic gaps and revealing reactions not yet described for the organism. Furthermore, M2N produces a graphical representation of the network indicating the association between the metabolite profile and predicted enzymes. This network allows identifying existing and new enzymatic reactions related to the organism of interest. The presence of false positives may lead you to improbable hypotheses. To avoid this, a thorough literature survey is recommended for further biological evidence indicating that enzymes predicted by M2N are present in the metabolism of the organism.

The results produced by M2N promote faster and better decisions about which enzymes are related to the input data and which of them should be included in metabolic networks. Doing so, M2N not only reduces the time spent for feeding metabolic networks using metabolomics data, but also generates important sources of knowledge. Additionally, with small modifications, M2N could easily be coupled to data sets generated by other omics
technologies, which may be able to integrate these data sets in a single work frame and improve further biological interpretation.

### 6.3. M2N applied to *Lactococcus lactis* IL1403

To illustrate the use of M2N, I analyzed the metabolomics data set generated in Chapter 5 (Table 6.2). This metabolomics data set consists of intra- and extracellular metabolite data of *Lactococcus lactis* IL1403 grown aerobically and anaerobically in Chemically Defined Medium (CDM) supplemented with either galactose, glucose or lactose + cellobiose. Metabolomics samples were harvested at middle to late exponential phase and processed as described by Smart et al., 2010.

In order to evaluate the results generated by M2N, I calculated the percentage of ECs predicted by M2N that are present in the metabolic network produced in Chapter 2 (positive matches). This network was reconstructed based on existing metabolic networks curated via an extensive literature review. Thus, the percentage of positive matches indicates the power of M2N in correctly predicting enzymes related to metabolite profiles. The higher the percentage of positive matches, the higher the power of M2N.

Negative matches might represent novel biochemical information about the metabolism of *L. lactis*. Therefore, I searched the biochemical databases KEGG and MetaCyc for biological indications that ECs predicted by M2N could be involved in the metabolism of *L. lactis*. MetaCyc is a web-based database containing species-specific biochemical data, including stoichiometric equations. The stoichiometric equations associated with those ECs showing biological evidences were then collected and added to the metabolic network generated in Chapter 2. Finally, to verify the potential of this network in predicting the growth of *L. lactis*, a set of simulations was performed using Flux Balance Analysis (FBA).
**Table 6.2: M2N Input Data.** This table shows the list of 52 metabolites used to illustrate and validate the method M2N.

<table>
<thead>
<tr>
<th>Name</th>
<th>kegg</th>
</tr>
</thead>
<tbody>
<tr>
<td>C16:0 acid MCF derivative (Palmitic acid)</td>
<td>C00249</td>
</tr>
<tr>
<td>C18:0 acid MCF derivative (Stearic acid)</td>
<td>C01530</td>
</tr>
<tr>
<td>C8:0 acid MCF derivative (Caprylic acid)</td>
<td>C06423</td>
</tr>
<tr>
<td>cis-Aconitic acid MCF derivative</td>
<td>C00417</td>
</tr>
<tr>
<td>Citraconic acid MCF derivative</td>
<td>C02226</td>
</tr>
<tr>
<td>Citric acid MCF derivative</td>
<td>C00158</td>
</tr>
<tr>
<td>Cysteine MCF derivative</td>
<td>C00097</td>
</tr>
<tr>
<td>Glutamic acid MCF derivative</td>
<td>C00025</td>
</tr>
<tr>
<td>Glycine MCF derivative</td>
<td>C00037</td>
</tr>
<tr>
<td>4-Aminobenzoic acid MCF derivative</td>
<td>C00568</td>
</tr>
<tr>
<td>Isoleucine MCF derivative</td>
<td>C00407</td>
</tr>
<tr>
<td>Itaconic acid MCF derivative</td>
<td>C00490</td>
</tr>
<tr>
<td>Lactic acid MCF derivative</td>
<td>C00186</td>
</tr>
<tr>
<td>Leucine MCF derivative</td>
<td>C00123</td>
</tr>
<tr>
<td>Lysine MCF derivative</td>
<td>C00047</td>
</tr>
<tr>
<td>Malonic acid MCF derivative</td>
<td>C00383</td>
</tr>
<tr>
<td>Methionine MCF derivative</td>
<td>C00073</td>
</tr>
<tr>
<td>Nicotinic acid MCF derivative</td>
<td>C00253</td>
</tr>
<tr>
<td>p-Toluic acid MCF derivative</td>
<td>C01454</td>
</tr>
<tr>
<td>Phenylalanine MCF derivative</td>
<td>C00079</td>
</tr>
</tbody>
</table>
Proline MCF derivative

Pyroglutamic acid MCF derivative

Quinic acid MCF derivative

Serine MCF derivative

Succinic acid MCF derivative

Threonine MCF derivative

Tryptophan MCF derivative

Tyrosine MCF derivative

Valine MCF derivative

Alanine MCF derivative

Aspartic acid MCF derivative

Benzoic acid MCF derivative

C10:0 acid MCF derivative (Capric acid)

C12:0 acid MCF derivative (Lauric acid)

C14:0 acid MCF derivative (Myristic acid)

C20:0 acid MCF derivative (Arachidic acid)

C22:0 acid MCF derivative (Behenic acid)

C6:0 acid MCF derivative (Capronic acid)

3-Hydroxybenzoic acid MCF derivative

Dodecane

N2-Acetyl-L-lysine MCF derivative

Tartaric acid MCF derivative

C00148

C01879

C00296

C00716

C00042

C00188

C00078

C00082

C00183

C00041

C00049

C00180

C01571

C02679

C06424

C06425

C08281

C01585

C00587

C08374

C12989

C00898
M2N was applied to the organism ‘illa’, which represents *L. lactis* IL1403 in KEGG database. BLAST searches were performed against the amino acid sequences in the *non-redundant (nr)* database using ‘*Lactococcus lactis*’ as Entrez. Entrez is an integrated database retrieval system that can be used for constraining BLAST searches (Acland et al., 2013). In this case, BLAST searches were limited to proteins from nr associated with *L. lactis*. BLAST hits showing e-values lower than $1 \times 10^{-10}$ were classified as ‘positive similarity’. The e-value represents the similarity level between amino acid sequences. More details about the databases used, the *Entrez* and the e-values are available at the NCBI website (http://blast.ncbi.nlm.nih.gov/).

### 6.3.1 M2N results for *L. lactis*

**Predicted ECs**

M2N identified 600 ECs, or enzymes, associated with the metabolite profiles produced in Chapter 5 (Table 6.2). Most of these, 483 ECs (80.5%), were not assigned to *L. lactis* IL1403 in KEGG database and presented no similar amino acid sequences for *L. lactis* in the protein

<table>
<thead>
<tr>
<th>Chemical Name</th>
<th>KEGG ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glutathione MCF derivative</td>
<td>C00051</td>
</tr>
<tr>
<td>Histidine MCF derivative</td>
<td>C00135</td>
</tr>
<tr>
<td>L-Norvaline MCF derivative</td>
<td>C01826</td>
</tr>
<tr>
<td>Heptadecane</td>
<td>C01816</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid MCF derivative</td>
<td>C01089</td>
</tr>
<tr>
<td>C18:1n-9 acid MCF derivative (Oleic acid)</td>
<td>C03425</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid MCF derivative</td>
<td>C00671</td>
</tr>
<tr>
<td>L-Ornithine MCF derivative</td>
<td>C00077</td>
</tr>
<tr>
<td>2-Phosphoenolpyruvic acid MCF derivative</td>
<td>C00074</td>
</tr>
<tr>
<td>C16:1n-7 acid MCF derivative (Palmitoleic acid)</td>
<td>C08362</td>
</tr>
</tbody>
</table>
database nr. This result was expected, since M2N searches for all the enzymes present in KEGG database, which contains enzymes associated with 2346 different organisms (http://www.genome.jp/kegg/catalog/org_list.html). Reapplying M2N using a different protein database, such as Swiss-prot, may result in additional enzymes showing similar protein sequences.

The other 117 ECs predicted by M2N were assigned to *L. lactis* IL1403 in KEGG database and/or returned positive hits from BLAST searches. These ECs represent enzymes shortlisted by M2N and stored in the file shortlist.csv. They were associated with 36 compounds from the input data (Table 6.3). Fourteen of these ECs were assigned to *L. lactis* IL1403 in KEGG database, while 91 were found in both KEGG database and also returned positive hits from BLAST searches. Twelve ECs showed positive hits from BLAST searches but were not assigned to *L. lactis* IL1403 in KEGG database, which might represent enzymes related to *L. lactis* but not yet described in the literature. Sixteen metabolites present in the input data showed no relation with any EC shortlisted by M2N.

Table 6.3. Shortlist. M2N shortlisted 117 EC numbers related to 36 metabolites in the input data and potentially playing part in the metabolism of *Lactococcus lactis*. The methods used to shortlist these EC numbers were: KEGG, if the EC number was assigned to *L. lactis* IL1403 in KEGG database; KEGG + BLAST, if the EC number was assigned to *L. lactis* IL1403 in KEGG database and also returned positive matches from BLAST searches; and BLAST, if the EC number was not assigned to *L. lactis* IL1403 in KEGG database but returned positive matches from BLAST searches.

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Method</th>
<th>Compound Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3.99.22</td>
<td>KEGG</td>
<td>Methionine</td>
</tr>
<tr>
<td>1.4.1.14</td>
<td>KEGG</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>1.5.1.24</td>
<td>KEGG</td>
<td>L-Ornithine</td>
</tr>
<tr>
<td>2.5.1.49</td>
<td>KEGG</td>
<td>Methionine/Cysteine</td>
</tr>
<tr>
<td>2.6.1.66</td>
<td>KEGG</td>
<td>Alanine/Valine</td>
</tr>
<tr>
<td>2.8.3.10</td>
<td>KEGG</td>
<td>Citric acid</td>
</tr>
<tr>
<td>3.5.1.4</td>
<td>KEGG</td>
<td>Benzoic acid</td>
</tr>
<tr>
<td>EC Number</td>
<td>Database</td>
<td>Name</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>4.1.3.38</td>
<td>KEGG</td>
<td>4-Aminobenzoic acid</td>
</tr>
<tr>
<td>1.1.1.27</td>
<td>KEGG</td>
<td>Lactic acid</td>
</tr>
<tr>
<td>4.2.1.35</td>
<td>KEGG</td>
<td>Citraconic acid</td>
</tr>
<tr>
<td>4.2.1.9</td>
<td>KEGG</td>
<td>3-Methyl-2-oxopentanoic acid</td>
</tr>
<tr>
<td>6.1.1.24</td>
<td>KEGG</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.5.6</td>
<td>KEGG</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.5.7</td>
<td>KEGG</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>1.1.1.23</td>
<td>KEGG + BLAST</td>
<td>Histidine</td>
</tr>
<tr>
<td>1.11.1.9</td>
<td>KEGG + BLAST</td>
<td>Glutathione</td>
</tr>
<tr>
<td>1.3.98.1</td>
<td>KEGG + BLAST</td>
<td>Succinic acid</td>
</tr>
<tr>
<td>1.3.99.1</td>
<td>KEGG + BLAST</td>
<td>Succinic acid</td>
</tr>
<tr>
<td>1.4.1.13</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>1.5.1.2</td>
<td>KEGG + BLAST</td>
<td>Proline</td>
</tr>
<tr>
<td>1.8.1.4</td>
<td>KEGG + BLAST</td>
<td>Glycine</td>
</tr>
<tr>
<td>1.8.1.7</td>
<td>KEGG + BLAST</td>
<td>Glutathione</td>
</tr>
<tr>
<td>1.8.1.9</td>
<td>KEGG + BLAST</td>
<td>Glutathione/Cysteine</td>
</tr>
<tr>
<td>1.8.4.11</td>
<td>KEGG + BLAST</td>
<td>Methionine</td>
</tr>
<tr>
<td>1.97.1.4</td>
<td>KEGG + BLAST</td>
<td>Methionine</td>
</tr>
<tr>
<td>2.1.1.14</td>
<td>KEGG + BLAST</td>
<td>Methionine</td>
</tr>
<tr>
<td>2.1.2.1</td>
<td>KEGG + BLAST</td>
<td>Glycine</td>
</tr>
<tr>
<td>2.1.3.2</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>2.1.3.3</td>
<td>KEGG + BLAST</td>
<td>L-Ornithine</td>
</tr>
<tr>
<td>2.3.1.1</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>2.3.1.35</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/L-Ornithine</td>
</tr>
<tr>
<td>2.3.3.1</td>
<td>KEGG + BLAST</td>
<td>Citric acid</td>
</tr>
</tbody>
</table>
2.4.2.1   KEGG + BLAST   Nicotinic acid
2.4.2.11  KEGG + BLAST   Nicotinic acid
2.4.2.14  KEGG + BLAST   Glutamic acid
2.5.1.15  KEGG + BLAST   4-Aminobenzoic acid
2.5.1.19  KEGG + BLAST   2-Phosphoenolpyruvic acid
2.5.1.47  KEGG + BLAST   Cysteine
2.5.1.48  KEGG + BLAST   Succinic acid/Cysteine
2.5.1.54  KEGG + BLAST   2-Phosphoenolpyruvic acid
2.5.1.6   KEGG + BLAST   Methionine
2.5.1.7   KEGG + BLAST   2-Phosphoenolpyruvic acid
2.6.1.11  KEGG + BLAST   Glutamic acid
2.6.1.16  KEGG + BLAST   Glutamic acid
2.6.1.2   KEGG + BLAST   Glutamic acid/Alanine
2.6.1.42  KEGG + BLAST   Methyl-2-oxopentanoic acid
2.6.1.52  KEGG + BLAST   Glutamic acid
2.6.1.85  KEGG + BLAST   Glutamic acid
2.6.1.9   KEGG + BLAST   Glutamic acid/Phenylalanine/Tyrosine
2.7.1.40  KEGG + BLAST   2-Phosphoenolpyruvic acid
2.7.2.11  KEGG + BLAST   Glutamic acid
2.7.2.4   KEGG + BLAST   Aspartic acid
2.7.3.9   KEGG + BLAST   2-Phosphoenolpyruvic acid
2.8.1.7   KEGG + BLAST   Alanine/Cysteine
3.4.11.2  KEGG + BLAST   Glycine/Cysteine
3.5.1.1   KEGG + BLAST   Aspartic acid
<table>
<thead>
<tr>
<th>EC Number</th>
<th>Database Method</th>
<th>Compound</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.1.16</td>
<td>KEGG + BLAST</td>
<td>L-Ornithine</td>
</tr>
<tr>
<td>3.5.1.18</td>
<td>KEGG + BLAST</td>
<td>Succinic acid</td>
</tr>
<tr>
<td>3.5.1.24</td>
<td>KEGG + BLAST</td>
<td>Glycine</td>
</tr>
<tr>
<td>3.6.1.7</td>
<td>KEGG + BLAST</td>
<td>Benzoic acid</td>
</tr>
<tr>
<td>4.1.1.15</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Aspartic acid</td>
</tr>
<tr>
<td>4.1.1.20</td>
<td>KEGG + BLAST</td>
<td>Lysine</td>
</tr>
<tr>
<td>4.1.3.27</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>4.1.3.6</td>
<td>KEGG + BLAST</td>
<td>Citric acid</td>
</tr>
<tr>
<td>4.2.1.11</td>
<td>KEGG + BLAST</td>
<td>2-Phosphoenoxyruvic acid</td>
</tr>
<tr>
<td>4.2.1.20</td>
<td>KEGG + BLAST</td>
<td>Tryptophan</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>KEGG + BLAST</td>
<td>Citric acid/cis-Aconitic acid</td>
</tr>
<tr>
<td>4.2.1.51</td>
<td>KEGG + BLAST</td>
<td>Phenylalanine</td>
</tr>
<tr>
<td>4.2.3.1</td>
<td>KEGG + BLAST</td>
<td>Threonine</td>
</tr>
<tr>
<td>4.3.1.19</td>
<td>KEGG + BLAST</td>
<td>Threonine</td>
</tr>
<tr>
<td>4.4.1.16</td>
<td>KEGG + BLAST</td>
<td>Alanine</td>
</tr>
<tr>
<td>4.4.1.8</td>
<td>KEGG + BLAST</td>
<td>Cysteine</td>
</tr>
<tr>
<td>5.1.1.1</td>
<td>KEGG + BLAST</td>
<td>Alanine</td>
</tr>
<tr>
<td>5.1.1.13</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>5.1.1.3</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.1.1.1</td>
<td>KEGG + BLAST</td>
<td>Tyrosine</td>
</tr>
<tr>
<td>6.1.1.10</td>
<td>KEGG + BLAST</td>
<td>Methionine</td>
</tr>
<tr>
<td>6.1.1.12</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>6.1.1.14</td>
<td>KEGG + BLAST</td>
<td>Glycine</td>
</tr>
<tr>
<td>6.1.1.15</td>
<td>KEGG + BLAST</td>
<td>Proline</td>
</tr>
<tr>
<td>6.1.1.16</td>
<td>KEGG + BLAST</td>
<td>Cysteine</td>
</tr>
<tr>
<td>Reaction ID</td>
<td>Method</td>
<td>Compound</td>
</tr>
<tr>
<td>------------</td>
<td>--------------</td>
<td>----------------</td>
</tr>
<tr>
<td>6.1.1.2</td>
<td>KEGG + BLAST</td>
<td>Tryptophan</td>
</tr>
<tr>
<td>6.1.1.20</td>
<td>KEGG + BLAST</td>
<td>Phenylalanine</td>
</tr>
<tr>
<td>6.1.1.21</td>
<td>KEGG + BLAST</td>
<td>Histidine</td>
</tr>
<tr>
<td>6.1.1.3</td>
<td>KEGG + BLAST</td>
<td>Threonine</td>
</tr>
<tr>
<td>6.1.1.4</td>
<td>KEGG + BLAST</td>
<td>Leucine</td>
</tr>
<tr>
<td>6.1.1.5</td>
<td>KEGG + BLAST</td>
<td>Isoleucine</td>
</tr>
<tr>
<td>6.1.1.6</td>
<td>KEGG + BLAST</td>
<td>Lysine</td>
</tr>
<tr>
<td>6.1.1.7</td>
<td>KEGG + BLAST</td>
<td>Alanine</td>
</tr>
<tr>
<td>6.1.1.9</td>
<td>KEGG + BLAST</td>
<td>Valine</td>
</tr>
<tr>
<td>6.3.1.2</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.2.12</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.2.17</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.2.5</td>
<td>KEGG + BLAST</td>
<td>Cysteine</td>
</tr>
<tr>
<td>6.3.2.6</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>6.3.2.8</td>
<td>KEGG + BLAST</td>
<td>Alanine</td>
</tr>
<tr>
<td>6.3.4.13</td>
<td>KEGG + BLAST</td>
<td>Glycine</td>
</tr>
<tr>
<td>6.3.4.19</td>
<td>KEGG + BLAST</td>
<td>Lysine</td>
</tr>
<tr>
<td>6.3.4.2</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.4.4</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>6.3.4.5</td>
<td>KEGG + BLAST</td>
<td>Aspartic acid</td>
</tr>
<tr>
<td>6.3.5.2</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.5.3</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>6.3.5.4</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Aspartic acid</td>
</tr>
<tr>
<td>6.3.5.5</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid</td>
</tr>
<tr>
<td>1.1.1.24</td>
<td>BLAST</td>
<td>Quinic acid</td>
</tr>
</tbody>
</table>
Positive matches

In order to evaluate the power of M2N in predicting enzymes, I calculated the percentage of predicted ECs that was part of the metabolic network presented in Chapter 2. This network was reconstructed based on the existing networks published by Oliveira et al. (2005) and Oddone et al. (2009). From 117 ECs predicted by M2N (Figure 6.2), 60% (70 ECs) were already present in the metabolic network introduced in Chapter 2, which indicates that more than half of the enzymes predicted by M2N are already considered to be related to *L. lactis*; 31% (36 ECs) were not present in the metabolic network of *L. lactis*, however, were described for *L. lactis* in both KEGG and MetaCyc. These ECs may be biochemical data generated after the reconstruction of the metabolic networks published by Oliveira et al. (2005) and Oddone et al. (2009); and 9% (11 ECs) were not described for *L. lactis* in MetaCyc, but were in KEGG database. Therefore, 91% of the enzymes predicted by M2N show indications of involvement in *L. lactis* metabolism.
Figure 6.2. Enzyme commission numbers (ECs) predicted by M2N and related to the metabolic network of *Lactococcus lactis*. Positive matches represent ECs predicted by M2N and present in the metabolic network presented in Chapter 2; Associated represents ECs not in this network, however, showing stoichiometric equations in MetaCyc specifically described for *L. lactis* IL1403; and Not associated represents ECs not present in the network and not related to *L. lactis* IL1403 in MetaCyc.

**Graphical outputs**

M2N generated 45 metabolic maps from KEGG database (Figure 6.3). KEGG metabolic maps are useful for highlighting the involvement of predicted ECs and metabolites in well-known metabolic pathways. This feature facilitates the identification of further reactions or missing steps in the conversion of enzymes predicted by M2N and enzymes described for *L. lactis* in KEGG database.
**Figure 6.3. Metabolic map generated by M2N.** M2N was applied to the metabolite profile presented in Chapter 5. As a result, M2N generated 45 metabolic maps from KEGG database highlighting metabolites in the input data and predicted EC numbers potentially playing part in the metabolism of *Lactococcus lactis* IL1403. This figure represents the metabolic pathway Alanine, Aspartate and Glutamate metabolism, where sharp rectangles represent enzymes; circles represent compounds and rounded rectangle represent metabolic pathways. Rectangles colored in green indicate enzymes assigned to *L. lactis* IL1403 in KEGG database but not predicted by M2N, while rectangles colored in yellow indicate enzymes predicted by M2N. Colored circles indicate compounds present in the input data.

In addition, M2N generated one metabolic network showing relationships between the 117 ECs shortlisted by M2N and the metabolites used as input data (Figure 6.4). This metabolic network shows 14 predicted ECs associated with at least two compounds from the input data (Table 6.5). These ECs represent potential reactions involving inter-conversion between these related compounds. After searching MetaCyc, 57% of these ECs (8) were confirmed as being involved with reactions catalyzing the conversion of metabolites reported by M2N. Further research on the other six ECs reported by M2N may generate additional knowledge about reactions not yet described for *L. lactis* IL1403.
Figure 6.4. Metabolic network produced by M2N. M2N was applied to the metabolite profile presented in Chapter 5. One of the results generated by M2N is a metabolic network connecting metabolites in the input data and the EC numbers predicted by this new method. EC numbers linked to two or more compounds indicate potential reactions involving the interconversion of these specific compounds.
Table 6.5. EC numbers representing potential reactions. Applied to the metabolomics data set generated in Chapter 5, M2N reported 14 EC numbers related to at least two compounds. The EC numbers in black/italic represent reactions described in the literature as involving the conversion of metabolites suggested by M2N.

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Method</th>
<th>Compound Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5.1.49</td>
<td>KEGG</td>
<td>Methionine/Cysteine</td>
</tr>
<tr>
<td>2.6.1.66</td>
<td>KEGG</td>
<td>Alanine/Valine</td>
</tr>
<tr>
<td>1.8.1.9</td>
<td>KEGG + BLAST</td>
<td>Glutathione/Cysteine</td>
</tr>
<tr>
<td>2.3.1.35</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/L-Ornithine</td>
</tr>
<tr>
<td>2.5.1.48</td>
<td>KEGG + BLAST</td>
<td>Succinic acid/Cysteine</td>
</tr>
<tr>
<td>2.6.1.2</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Alanine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Glutamic acid/Leucine/Valine/Isoleucine/3-Methyl-2-oxopentanoic acid</td>
</tr>
<tr>
<td>2.6.1.42</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Phenylalanine/Tyrosine</td>
</tr>
<tr>
<td>2.6.1.9</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Cysteine</td>
</tr>
<tr>
<td>2.8.1.7</td>
<td>KEGG + BLAST</td>
<td>Glycine/Cysteine</td>
</tr>
<tr>
<td>3.4.11.2</td>
<td>KEGG + BLAST</td>
<td>Glycine/Cysteine</td>
</tr>
<tr>
<td>4.1.1.15</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Aspartic acid</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>KEGG + BLAST</td>
<td>Citric acid/cis-Aconitic acid</td>
</tr>
<tr>
<td>6.3.5.4</td>
<td>KEGG + BLAST</td>
<td>Glutamic acid/Aspartic acid</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C16:0 acid (Palmitic acid)/C18:0 acid (Stearic acid)/C12:0 acid (Lauric acid)/C14:0 acid (Myristic acid)/C16:1n-7</td>
</tr>
<tr>
<td>3.1.2.14</td>
<td>BLAST</td>
<td>acid (Palmitoleic acid)</td>
</tr>
</tbody>
</table>
Feeding L. lactis’ metabolic network

The results generated by M2N showed that 117 ECs were potentially related to the metabolism of L. lactis. These enzymes were then used to expand the metabolic network presented in Chapter 2. A high-quality metabolic network requires extensive literature survey to confirm the existence of each reaction in the network. Here, the objective is to evaluate if the results produced by M2N can be quickly used to expand metabolic networks. Thus, only MetaCyc was searched to find stoichiometric equations associated with predicted ECs and described specifically for L. lactis IL1403 and findGaps (Chapter 2) was used to assess potential metabolic gaps. As a result, 23 new reactions were added to the metabolic network of L. lactis (Table 6.6). These reactions were associated with 11 ECs predicted by M2N, and most of them are related with the biosynthesis of fatty acids. Thirty-two ECs were described in MetaCyc as being involved in protein complexes as reactions intermediates. Only 11 ECs predicted by M2N showed no stoichiometric equations associated with L. lactis, which may represent reactions not yet described for these bacteria.

Table 6.6. Reactions added to the genome-scale metabolic network of Lactococcus lactis.

<table>
<thead>
<tr>
<th>Enzyme</th>
<th>Reaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.5.5</td>
<td>2 ATP + L-glutamine + HCO(3) + H(2)O =&gt; 2 ADP + phosphate + L-glutamate + carbamoyl phosphate + 4 H(+)</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>cis-aconitate + H(2)O =&gt; isocitrate</td>
</tr>
<tr>
<td>4.2.1.3</td>
<td>citrate =&gt; cis-aconitate + H(2)O</td>
</tr>
<tr>
<td>2.5.1.48</td>
<td>NH(3) + 2-oxobutanoate + succinate + 2 H(+) &lt;=&gt; O-succinyl-L-homoserine + H(2)O</td>
</tr>
<tr>
<td>2.4.2.1</td>
<td>deoxyadenosine + phosphate =&gt; adenine + 2-deoxy-D-ribose 1-phosphate</td>
</tr>
<tr>
<td>2.4.2.1</td>
<td>adenosine + phosphate =&gt; adenine + D-ribose 1-phosphate</td>
</tr>
<tr>
<td>1.3.98.1</td>
<td>(S)-dihydroorotate + fumarate =&gt; orotate + succinate</td>
</tr>
<tr>
<td>1.1.1.23</td>
<td>L-histidinol + NAD(+) =&gt; L-histidinal + NADH + H(+)</td>
</tr>
<tr>
<td>1.1.1.23</td>
<td>L-histidinal + NAD(+) + H(2)O =&gt; L-histidine + NADH + H(+)</td>
</tr>
<tr>
<td>4.1.3.38</td>
<td>chorismate + L-glutamine =&gt; 4-amino-4-deoxychorismate + L-glutamate</td>
</tr>
<tr>
<td>4.1.3.38</td>
<td>4-amino-4-deoxychorismate =&gt; 4-aminobenzoate + Pyruvate</td>
</tr>
</tbody>
</table>
3.1.2.14 hexadecanoyl-(acp) + H(2)O ⇄ hexadecanoate + ACP
3.1.2.14 hexadecanoyl-(acp) + O(2) ⇄ hexadecenoyl-(acp) + 2 H(2)O
3.1.2.14 hexadecenoyl-(acp) + H(2)O ⇄ hexadecenoate + ACP
3.1.2.14 Hexadecanoyl-(acp) + Malonyl-(acp) ⇄ 3-oxostearoyl-(acp) + CO(2) + ACP
3.1.2.14 3-oxostearoyl-(acp) + NADPH + H(+) ⇄ (R)-3-hydroxyoctadecanoyl-(acp) + NADP(+)
3.1.2.14 (R)-3-hydroxyoctadecanoyl-(acp) ⇄ trans-octadec-2-enoyl-(acp) + H(2)O
3.1.2.14 trans-octadec-2-enoyl-(acp) + NADH + H(+) ⇄ octadecanoyl-(acp) + NAD(+)
3.1.2.14 octadecanoyl-(acp) + H(2)O ⇄ octadecanoate + ACP
3.1.2.14 octadecanoyl-(acp) + O(2) ⇄ octadecenoyl-(acp) + 2 H(2)O
3.1.2.14 octadecenoyl-(acp) + H(2)O ⇄ octadecenoate + ACP
3.1.2.14 tetradecanoyl-(acp) + H(2)O ⇄ tetradecanoate + ACP
3.1.2.14 dodecanoyl-(acp) + H(2)O ⇄ dodecanoate + ACP

Expanded metabolic network

The metabolic network of L. lactis introduced in Chapter 2 contained a total of 921 reactions (Appendix 2j), where 154 were involved in transport, 204 in compartmental exchange and 563 occurred intracellularly. Based on the metabolite profile generated in Chapter 5, M2N was able to expand the network to 951 reactions, by adding one to transport, eight to exchange and 22 to intracellular reactions (see Appendix 6a). Importantly, this new metabolic network contains 12 more reactions involving the biosynthesis of fatty acids.

In order to qualitatively evaluate the potential of this expanded metabolic network in modeling L. lactis’ metabolism, Flux Balance Analysis (FBA) was applied for simulating biomass production using glucose as the single carbon source in anaerobic conditions. For this, the biosynthesis of alanine was blocked and its uptake rate constrained to 0.573 mmolgDW⁻¹h⁻¹, the same value previously used by Oliveira et al. (2005) (see Appendix 6b for flux constrains). As alanine is an essential amino acid, increasing the uptake rate of glucose was expected to increase biomass production until a point where alanine becomes a
limiting factor. In this case, the rate of biomass production is expected to remain constant, despite of further increments in glucose uptake rate.

Figure 6.5 shows the results from FBA simulations performed using the metabolic network produced here. It shows the changes in biomass production in response to changes in the uptake rates of glucose. The maximum rate of biomass production was achieved at a glucose uptake rate of 200.6 mmol gDW\(^{-1}\) h\(^{-1}\). Increasing the uptake rate of glucose resulted in no increase in biomass production rate, which indicates that alanine was actually a potential limiting factor. The values of glucose uptake rate and the rate of biomass production are certainly not in agreement with the literature (Oliveira et al., 2005). Further tuning of this metabolic network is essential to perform accurate predictions about the metabolism of \(L.\) \textit{lactis}. Moreover, varying levels of gene expression would be represented in different fluxes through the reactions associated to these genes, where a higher gene expression would generally result in a higher flux through associated reactions. We would need to know the value of this change in flux to be able to incorporate this information in the metabolic network and have it considered in future simulations. Although the aim of this chapter was not to generate a well-curated genome-scale model, the qualitative modeling of \(L.\) \textit{lactis}' growth already indicates a great potential of the metabolic network produced here in modeling the metabolism of this bacterium.

![Figure 6.5. Flux balance analysis applied for anaerobic growth when constraining alanine availability and maximizing for biomass formation. Growth rate is plotted against glucose uptake rate. Model results are from simulations S1 to S6 (see Appendix 6b).](image-url)
In summary, based solely in the metabolite profiles produced in Chapter 5, M2N predicted 117 enzymes potentially involved in the metabolism of *L. lactis* IL1403. Sixty percent of these were already present in the metabolic network of *L. lactis* produced in Chapter 2, while 31% were described for this bacterium in the species-specific database MetaCyc. In total, 91% of the enzymes predicted by M2N were already confirmed as being involved with *L. lactis* and had potential to be included in its metabolic network. Additionally, M2N generated 45 metabolic maps highlighting predicted enzymes and a metabolic network revealing potential reactions involving metabolites in the input data. Using M2N results, the metabolic network produced in Chapter 2 was expanded and its analysis using FBA demonstrated that it has potential for modeling the metabolism of *L. lactis*.

### 6.4. M2N Toolbox

The M2N algorithm was implemented in MATLAB as M2N Toolbox. The code of every function in M2N Toolbox is available in Appendix 6. M2N Toolbox depends on loadcell.m (http://www.mathworks.com/matlabcentral/fileexchange/1965-loadcell-m) and comprises seven functions: install_blast, databases, install_databases, MET2EC, EC2BLAST, BLAST2MAPS and BLAST2NET. M2N Toolbox allows users to perform BLAST comparisons through the Internet or using a local BLAST installation. The function install_blast is used to install BLAST software locally in case users have no previous BLAST installation. Before performing a new installation, install_blast searches for an existing BLAST installation and, if found, it tests its functionality and fixes potential errors. Local BLAST comparisons require the installation of databases such as *nr* and Swiss-Prot. The function databases is then used to visualize databases already installed locally, while install_databases is used to locally install new databases from the Internet or from a local file. MET2EC is used to obtain the EC numbers associated with each metabolite present in the input data from KEGG database. It subsequently generates a CSV file (met2ec.csv) containing the metabolites in the input data and their respective EC numbers. The result from MET2EC can be directly used as input data for the function EC2BLAST, which searches for the protein sequence of each enzyme and compares it against the desired protein database. Then, it proceeds to generate two CSV files: full_report.csv and shortlist.csv. The file full_report.csv contains all the BLAST comparisons while the shortlist.csv contains only enzymes that returned positive matches. The CSV files produced by EC2BLAST are then
used as input data for the functions BLAST2MAPS and BLAST2NET to generate metabolic maps and a metabolic network, respectively.

Functions:

- **install_blast()**

  install_blast (see Appendix 6c) was developed to install BLAST software locally. Firstly, it searches for an existent BLAST installation. If BLAST is not installed, then it installs BLAST using the most recent installation file available at NCBI server. Alternatively, the user may point to a local installation file. This function has no arguments to be edited by the user. Every required information and decision is taken by pop up dialog boxes.

- **install_database()**

  install_database (see Appendix 6d) is used to locally install protein databases for BLASTing. This function allows the user to install databases from the NCBI server or from local files containing the database compacted in tar.gz format. Database files are available at ftp://ftp.ncbi.nlm.nih.gov/blast/db/. This function requires no arguments to be manually edited by the user. Every required information and decision is taken by pop up dialog boxes.

- **databases()**

  databases (see Appendix 6e) is used for searching currently installed protein databases. This function allows the user to verify which protein databases are locally installed. This function requires no arguments to be manually edited by the user. Every required information and decision is taken by pop up dialog boxes.

- **MET2EC()**

  MET2EC (see Appendix 6f) is used for collecting from the KEGG database EC numbers related to metabolite profiles. As input data, it uses a CSV file containing names of metabolites in the first column and their respective KEGG codes in the second column (see Table 6.1). This function requires no arguments to be manually edited by the user. Every
required information and decision is taken by pop up dialog boxes. MET2EC generates a CSV file containing EC numbers added to the input data.

- **EC2BLAST()**

EC2BLAST (see Appendix 6g) performs two tasks: it collects the amino acid sequences associated with EC numbers from the European Bioinformatics Institute (EBI), and compares these sequences against a protein database using BLAST. As input data, it uses the CSV file generated by MET2EC, the KEGG code for the organism studied, the maximum \(e\)-value to be used as threshold and the \(entrez\). The KEGG code for the organism studied can be found in http://www.genome.jp/kegg/catalog/org_list.html. The values to be used as \(e\)-value and \(entrez\) can be found in http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=FAQ. This function requires no arguments to be manually edited by the user. Every required information and decision is taken by pop up dialog boxes. EC2BLAST generates two CSV files: full_report.csv, which contains all the information in the input data with the addition of the results generated by every BLAST comparison; and shortlist.csv, which contains the results only for EC numbers showing association with the organism studied and/or positive similarity returned from BLAST searches.

- **BLAST2MAPS()**

BLAST2MAPS (see Appendix 6h) is used for collecting from the KEGG database maps of metabolic pathways related to predicted ECs and metabolites in the input data. As input data, it uses the files full_report.csv or shortlist.csv previously generated by EC2BLAST. This function requires the arguments bg_color and text_color to be manually edited by the user. These arguments define the color used to highlight predicted enzymes and compounds. BLAST2MAPS creates a folder called pathway_maps where the PNG file referent to each metabolic pathway is then saved. The KEGG code of the organism under study is prompted to the user through a popup dialog box.

- **BLAST2NET()**

BLAST2NET (see Appendix 6i) is used for generating a metabolic network using the files full_report.csv or shortlist.csv. This function requires no arguments to be manually edited by the user. Every required information and decision is taken by pop up dialog boxes.
BLAST2NET generates a metabolic network within the MATLAB environment. The user is then able to manually customize the network according to his requirements.

6.5. Conclusion

In this chapter, I introduced a new algorithm, M2N, developed to facilitate the use of metabolomics data for feeding metabolic networks. It uses simple metabolite profiles, freely accessible databases and BLAST to predict enzymes, or EC numbers, potentially related to a specific organism. I illustrated the use of M2N by analyzing a metabolite profile generated for the bacterium *L. lactis* containing 52 metabolites (Chapter 5). As a result, 91% of the enzymes predicted by M2N showed actual evidence of involvement in the metabolism of this bacterium, which indicates a great performance of M2N as a discovery tool. In addition, although *L. lactis* is a very well studied organism, M2N was able to detect a new enzyme associated with biosynthesis of fatty acids not yet described in KEGG database. The graphical outputs generated by M2N were able to predict reactions linking metabolites in the input data and assisted the expansion of the metabolic network produced in Chapter 2. This metabolic network has now 23 new reactions and its simulations using FBA indicated that it is able to qualitatively predict the growth of *L. lactis*. M2N considerably improves the time spent in using metabolomics data for feeding metabolic networks. To the best of my knowledge, this is the first tool able to use simple metabolite profiles for feeding metabolic networks. Expanding M2N to be capable of including transcriptomics and proteomics data may allow the integration of these *omics*-related technologies, and increase knowledge-generation and facilitate data interpretation. Lastly, M2N was implemented in a MATLAB Toolbox using pop-dialog boxes, which facilitates its usage by new MATLAB users.
Chapter 7. Summarizing discussion and future directions.
7.1. Summarizing discussion

Throughout this thesis, I presented new methods, algorithms and bioinformatics tools that address some of the limitations associated with metabolomics and systems biology studies. The lactic acid bacteria *L. lactis* was used as a model organism to illustrate the potential of each of the methods and algorithms presented here because of the extensive information available about its metabolism, its economic importance, the availability of its annotated genome sequence and two existing genome-scale metabolic networks.

In order to explore and expand the capabilities of the genome-scale metabolic networks reconstructed for *L. lactis*, a method using its annotated genome sequence and FBA was proposed and applied. This method was able to semi-automatically merge the content of these networks and find metabolic gaps (findGaps). As a result, it produced a genome-scale metabolic network containing 921 reactions and 668 metabolites.

Further expanding the metabolic network of *L. lactis* required experimental data, in this case, produced via metabolomics. Although the analytical aspects of metabolomics are well developed, the same cannot be said about the processing of GC-MS-generated data. AMDIS, the most popular software in metabolomics for deconvoluting GC-MS chromatograms and identify metabolites, requires extensive manual curation, which introduces variability in the data due to human error. Furthermore, such manual curation is tremendously time-consuming. A new algorithm, MetaBox, which deconvolutes GC-MS chromatograms, and identifies and quantifies metabolites, was therefore proposed. The performance of this algorithm was evaluated against AMDIS. MetaBox considerably reduced the number of false positives reported by AMDIS and generated more accurate quantifications.

Implemented in an R-software package, MetaBox automatically generates results in a format that facilitates further data processing. The functions in the MetaBox R package were specifically developed to normalize and perform basic statistical analyses to metabolite’s abundances across experimental conditions, thereby allowing high-throughput investigation of metabolomics data in a flexible manner. These functions constitute the main pipeline used to process almost every single metabolomics sample analyzed at The University of Auckland. Furthermore, 57 institutions from more than 15 different countries have already requested to use these tools.
Due to the convoluted nature of cell metabolism and the complex regulatory processes resulting in different levels of metabolites between experimental conditions, metabolomics data is considered one of the most difficult omics data to interpret. A new method, PAPi, was subsequently developed to assist in metabolomics data interpretation by predicting the activity of metabolic pathways. PAPi uses the number of metabolites identified and their respective levels to calculate the activity score associated with metabolic pathways. Although PAPi does not calculate the actual flux through metabolic pathways, it enables comparisons of predicted pathway activities between different experimental conditions, which ultimately results in hypotheses to assist data interpretation and to determine the direction of future experiments. PAPi has been successfully applied to identify metabolic mechanisms driving the morphology of Candida albicans (Han et al., 2012), metabolic changes in yeast metabolism in response to different sound stimuli (Aggio et al., 2011) and key metabolic pathways in bacteria associated with environmental stress responses (unpublished data).

Using MetaBox for processing metabolomics data and PAPi for predicting the activity of metabolic pathways, metabolite profiles of L. lactis IL1403 were produced and analyzed to further expand the metabolic network of these bacteria. By subjecting L. lactis to different carbon sources and oxygen availabilities I was able to identify a broader range of metabolites than what was previously available. As a result, 58 metabolites were identified as being part of the metabolism of L. lactis IL1403 and novel biological hypotheses were generated. For instance, this was the first study suggesting the ability of L. lactis to undergo respiration in the presence of cyanocobalamin and the first study suggesting a mechanism for the excretion of proline in these bacteria. Most hypotheses that resulted from these experiments are of great interest to the dairy industry as they may be used for improving the final properties of fermented food products.

Finally, the metabolite profiles of L. lactis IL1403 were used to expand the reconstructed metabolic network of these bacteria. Although metabolomics has been widely applied to systems biology studies, there is no method able to use metabolite profiles for feeding metabolic networks. Hence, a new method called M2N was developed, and used to feed the metabolic network of L. lactis. M2N makes use of simple metabolite profiles to predict enzymes potentially related to the organism under study. Using this method, I was able to shortlist 117 enzymes (ECs) that, according to two major biochemical databases (KEGG and MetaCyc) and BLAST searches, are potentially associated with L. lactis and with the metabolites contained in the profile used. Ninety-one percent of these enzymes had already
been demonstrated to have involvement in the metabolism of these bacteria, which supports the great prediction power of this new method. In addition, M2N produces graphical outputs that facilitated the use of predicted enzymes for expanding metabolic networks, and showed great potential in predicting reactions linking metabolites from the input data. Based on M2N outputs, 23 reactions were added to the network of *L. lactis*, which now contains 951 reactions and 670 metabolites. Growth simulations using FBA indicated that the new network has great potential in modeling the metabolism of these bacteria.

In summary, the algorithms and methods presented in this thesis improve metabolomics and systems biology studies by: 1) semi-automating the reconstruction of genome-scale metabolic networks based on existent networks, 2) processing metabolomics data in a more reliable and high-throughput manner, 3) generating hypotheses to assist the biological interpretation of metabolomics data and 4) facilitating the use of metabolomics data for feeding metabolic networks. The use of *L. lactis* as the model organism for illustrating and validating these tools resulted in novel hypotheses and an expanded genome-scale metabolic network of this bacterium.

With the ever-increasing technologies for data generation, bioinformatics became vital to keep up with the amount of and give meaning to the information produced. The algorithms and computational tools presented in this thesis represent a step forward in the evolution of the field of bioinformatics, specifically for metabolomics. These tools, as compared to the commonly used approaches, considerably increase the efficiency and quality of data processing and analysis in systems biology and metabolomics studies. Ultimately, the use of these tools implicates fewer resources required for fully processing experiments, and increases the reliability of the final biological interpretation.
7.2. Future directions

META BOX

MetaBox was developed to identify and quantify metabolites analyzed by GC-MS. When compared to AMDIS, MetaBox demonstrated great improvement in identification and better reproducibility associated with metabolite quantification. Still, there is certainly room for improvement. Further enhancement of the MetaBox algorithm, such as using additional statistical methods to compare fragmentation patterns, may produce even better results. Although MetaBox shows great improvement in relation to AMDIS, it requires basic knowledge of R language. In order to become a common tool and compete in the global market with freeware such as AMDIS, MetaBox should be coupled with a user-friendly interface that requires no previous knowledge of computer code.

PATHWAY ACTIVITY PROFILING - PAPi

PAPi was developed to assist in the biological interpretation of metabolite profiles. Although PAPi has been successfully applied to many metabolomics studies, further comparisons of PAPi’s predictions against results from fluxomics experiments would greatly improve the assumptions behind this method. In addition, as KEGG database contains data from other omics technologies, PAPi can be extended so that it can integrate various omics results in a single framework. This way, PAPi would generate information that could potentially lead to better understanding on a system-wide level. As with MetaBox, PAPi requires an improved user-friendly interface to become a common tool used by the metabolomics community.

METABOLIC NETWORK OF L. LACTIS

The genome-scale metabolic network of L. lactis presented in this thesis was reconstructed mainly to illustrate the use of new methods for reconstructing and curating metabolic networks. Although growth simulations applied to this network showed great potential in modeling the metabolism of L. lactis, tuning is necessary to quantitatively fit the simulations to experimental data. Further research on the results generated by M2N may generate additional knowledge about L. lactis’ metabolism and add biological features to this metabolic network.
References


Investigation of the adaptation of Lactococcus lactis to isoleucine starvation integrating dynamic transcriptome and proteome information. Microbial Cell Factories, 10, 11. doi: 10.1186/1475-2859-10-s1-s18


and metabolite formation under aerobic and anaerobic conditions in the presence or absence of hemin. Biotechnology and Bioengineering, 95(6), 1070-1080. doi: 10.1002/bit.21070


control of the Lactococcus lactis trp Operon. Journal of Bacteriology, 180(12), 3174-3180.


Appendices

Appendices numbers correspond to their respective chapter number.

Appendix 2a.

R script MergeNetworks for Mac OS.

```r
library(tcltk)

## Upload metabolic network 1 (File 1 - supplemental material) as a vector called "net1"
net1 <- read.csv(tk_choose.files(), sepal=",")
names(net1)[1:3] <- c("RXNID", "net1.equation", "net1.subsystem")

## Upload metabolic network 2 (File 2 - supplemental material) as a vector called "net2"
net2 <- read.csv(tk_choose.files(), sepal=",")

## Merging net1 and net2.
net12 <- merge(net1, net2, by.x="RXNID", by.y="RXNID", all=TRUE)
net12$net1.equation <- as.character(net12$net1.equation)
net12$net1.equation <- gsub("<", "<=" , net12$net1.equation)
net12$net1.equation <- gsub("<", "<=>", net12$net1.equation)
net12$net1.equation <- gsub(">", "=>", net12$net1.equation)
net12$net2.equation <- as.character(net12$net2.equation)
net12$net2.equation <- gsub("<", "<=" , net12$net2.equation)
net12$net2.equation <- gsub("<", "<=>", net12$net2.equation)
net12$net2.equation <- gsub(">", "=>", net12$net2.equation)

for (i in 1:nrow(net12)){
  if (net12$net1.equation[i] == 0 || net12$net2.equation[i] == 0){
    if (net12$net1.equation[i] == 0){
      net12$fixed[i] <- as.character(net12$net2.equation[i])
      net12$correct[i] <- "YES"
    } else {
      net12$fixed[i] <- net12$net1.equation[i]
      net12$correct[i] <- "YES"
    }
  } else {
    if(as.character(net12$net1.equation[i]) == as.character(net12$net2.equation[i])){
      net12$correct[i] <- "YES"
    } else {
      net12$correct[i] <- "NO"
    }
  }
}

net12$correct <- as.character(net12$correct)
net12$fixed <- as.character(net12$fixed)

# result_script1

write.csv(net12, file = store, row.names = FALSE)

```

Appendix 2b.

R script addAGS for Mac OS.

```r
library(tcltk)

## Upload the file containing the annotated genome sequence of L.lactis as gene.seq (File 3 - supplemental material). The file must be in .CSV format.

```
gene.seq <- read.csv(tk_choose.files(), sep=";
## Keep only columns containing the gene loccus, gene ID and description.
gene.seq <- gene.seq[c(1,2,3)]
## Rename the columns to "loccus", "geneid" and "description".
names(gene.seq)[1:3] <- c("loccus", "geneid", "description")
## Create and names new columns to allocate the information from net12.
gene.seq$net12.geneid <- "-" * 3
gene.seq$net12.equation <- "-" * 3
gene.seq$net12.subsystem <- "-" * 3
## Collect the data from net12 and insert in gene.seq.
for (i in 1:nrow(gene.seq)){
cross_ref <- grep(as.character(gene.seq$geneid[i]), as.character(net12$RXNID), value=TRUE)
  if (length(cross_ref) == 1){
    net12.row <- subset(net12, net12$RXNID == cross_ref)
    gene.seq$net12.geneid[i] <- as.character(net12.row$RXNID)
    gene.seq$net12.equation[i] <- as.character(net12.row$fixed)
    gene.seq$net12.subsystem[i] <- paste(as.character(net12.row$net1.subsystem),
                                          as.character(net12.row$net2.subsystem))
  } else {
    if (length(cross_ref) > 1){
      gene.seq$net12.geneid[i] <- "more_than_one"
      gene.seq$net12.equation[i] <- "more_than_one"
      gene.seq$net12.subsystem[i] <- "more_than_one"
      net12.row <- subset(net12, net12$RXNID == cross_ref[1])
      for (k in 2:length(cross_ref)){
        net12.row <- rbind(net12.row, (subset(net12, net12$RXNID == cross_ref[k])))
      }
    }
    for (j in 1:length(cross_ref)){
      gene.seq <- rbind(gene.seq, data.frame(loccus=gene.seq$loccus[i],
                                              geneid=net12.row$RXNID[j],
                                              description=gene.seq$description[i],
                                              net12.geneid=net12.row$RXNID[j],
                                              net12.equation=net12.row$fixed[j],
                                              net12.subsystem=paste(as.character(net12.row$net1.subsystem[j]),
                                                        as.character(net12.row$net2.subsystem[j])))
    }
  }
}
}
sheet <- "result_script2"
folder = tk_choose.dir()
store <- paste(folder, "/", sheet, ".csv", sep = "")
write.csv(gene.seq, file = store, row.names = FALSE)

############################ THE END ############################

Appendix 2c.
R script getECNumbers for Mac OS.

### Step 3 - Collecting the EC numbers present in the column "gene.seq$description" and inserting it in a new column called "ec.number".

####
gene.seq$ec.number <- "-" * 3
for (i in 1:nrow(gene.seq)){
  part.of <- strsplit(as.character(gene.seq$description[i]), " ")
  if (length(part.of) > 0){
    part.of <- data.frame(part.of)
    for (j in 1:nrow(part.of)){
      ec.found <- grep("EC", part.of[j,], value=TRUE, fixed = TRUE)
      if (length(ec.found) > 0){
        ec.number <- paste(ec.found, ec.found + 1, sep="")
        print(ec.number)
      }
    }
  }
}
sheet <- "result_script3"
folder = tk_choose.dir()
store <- paste(folder, "/", sheet, ".csv", sep = "")
write.csv(gene.seq, file = store, row.names = FALSE)

#################################################################### THE END #
Appendix 2d.

R script getECKEGG for Mac OS.

```R
### Check how to organize the EC numbers with the hifen in between numbers! ###
### Step 4 - Collecting more EC numbers from KEGG database using the loccus id from gene.seq.
kegg.list <- read.csv(tk_choose.files(), sep=""

#### Select enzymes where the EC is complete to the enzyme ####
kegg.list <- kegg.list[(grep("-", kegg.list$Sec.list)),]

#### Delete the "lla:" from kegg.list ####
kegg.list$llactis.locus <- gsub("lla:" "", kegg.list$llactis.locus)

#### Add in llactis.jcvi the enzyme numbers from kegg.list, but only geneIDs with no number ####
for (i in 1:nrow(gene.seq)){
  check.ec <- grep("-", gene.seq$Sec.number[i], value = TRUE)
  if (length(check.ec) == 0){
  } else {
    check.ec2 <- subset(kegg.list, kegg.list$llactis.locus == gene.seq$locus[i])
    if (nrow(check.ec2) > 0){
      gene.seq$Sec.number[i] <- gsub("ec:" "",
      as.character(check.ec2$Sec.list[1]))
    } else {
      gene.seq$Sec.number[i] <- gsub("ec:" "",
      as.character(paste(check.ec2$Sec.list, collapse="@")))
    }
  }
}
sheet <- "result_script4"
folder = tk_choose.dir()
store <- paste(folder, "/", sheet, ".csv", sep = "")
write.csv(gene.seq, file = store, row.names = FALSE)

######################## THE END ###############################
```

Appendix 2e.

R script getReactions for Mac OS.

```R
### Step 5 - Collecting potential reactions from KEGG and their respective KEGG identifiers for reactions and compounds ###
import kegg reactions file (File 12 - Appendix) ###
kegg.reaction <- tk_choose.files()
kegg.reaction <- readLines(kegg.reaction)

### getting the reactions from kegg and includ in JCVI ###
gene.seq$kegg.equation <- "-" gene.seq$kegg.equation.codes <- "-" gene.seq$kegg.reaction.id <- "-" gene.seq$Correct <- "-" gene.seq$Fixed <- "-"
gene.seq$number[949] = "5.4.2.1"
for (i in 1:nrow(gene.seq)){
  print(i)
  row.on <- as.character(gene.seq$Sec.number[i])
  if (row.on == "-"副){
    in.kegg <- grep(paste(" ",row.on), kegg.reaction, value=FALSE)
    if(length(in.kegg) == 0){
      in.kegg <- grep("Sampa", "rio", value=FALSE)
    }
    if(length(in.kegg) == 1){
      definition <- grep("DEFINITION", kegg.reaction[(in.kegg)-scan.row]), value=TRUE)
      while (length(definition) == 0){
        definition <- grep("DESCRIPTION", kegg.reaction[[in.kegg]-scan.row]), value=TRUE)
        if(length(definition) > 0){
          extra.row <- grep("EQUATION", kegg.reaction[(in.kegg[j])-(scan.row-1)])
          if(length(extra.row) > 0){
            } else {
```

169
definition <- paste(definition, gsub("","", kegg.reaction[((in.kegg[j])-(scan.row-1))]), fixed=TRUE), sep=" ")
}
else {
}
scan.row <- (scan.row + 1)
}
definition <- gsub("DEFINITION   ", ",", definition, fixed=TRUE)
gene.seq$kegg.equation[i] <- definition

get equation with kegg code cpd ######
scan.row <-0
definition <- grep("sampa", "rio", value=TRUE)
while (length(definition) == 0){
definition <- grep("EQUATION", kegg.reaction((in.kegg)-scan.row)), value=TRUE)
scan.row <- (scan.row + 1)
}
definition <- gsub("EQUATION    ", ",", definition, fixed=TRUE)
gene.seq$kegg.equation.codes[i] <- definition
definition <- strsplit(definition, ")
definition <- data.frame(definition)

get reaction ID ######
scan.row <-0
definition <- grep("sampa", "rio", value=TRUE)
while (length(definition) == 0){
definition <- grep("ENTRY", kegg.reaction((in.kegg)-scan.row)), value=TRUE)
scan.row <- (scan.row + 1)
}
definition <- gsub("ENTRY    ", ",", definition, fixed=TRUE)
definition <- gsub("EQUATION    ", ",", definition, fixed=TRUE)
gene.seq$kegg.reaction.id[i] <- definition

Verify if net12.equation is equal to kegg.equation

if (as.character(gene.seq$net12.equation[i]) == as.character(gene.seq$kegg.equation[i])){
gene.seq$correct <- "YES"
} else {

gene.seq$correct <- "NO"
}
}
divisor <- gene.seq$net12.equation[i]
if (divisor == "+-*
for (g in 1){
scan.row <-0
definition <- grep("sampa", "rio", value=TRUE)
while (length(definition) == 0){
definition <- grep("EQUATION", kegg.reaction((in.kegg)-(scan.row-1))), value=TRUE)
if (length(definition) > 0){
extra.row <- grep("EQUATION", kegg.reaction((in.kegg)-(scan.row-1)))
if(length(extra.row) > 0){
definition <- paste(definition, gsub("","", kegg.reaction((in.kegg)-(scan.row-1))), fixed=TRUE), sep=" ")
}
scan.row <- (scan.row + 1)
}
definition <- gsub("DEFINITION   ", ",", definition, fixed=TRUE)
gene.seq$kegg.equation[i] <- definition
scan.row <-0
definition <- grep("sampa", "rio", value=TRUE)
while (length(definition) == 0){
definition <- grep("EQUATION", kegg.reaction((in.kegg)-(scan.row-1))), value=TRUE)
scan.row <- (scan.row + 1)
}
definition <- gsub("EQUATION    ", ",", definition, fixed=TRUE)
gene.seq$kegg.equation.codes[i] <- definition
definition <- strsplit(definition, ")
definition <- data.frame(definition)

get reaction ID ######
scan.row <-0
definition <- grep("sampa", "rio", value=TRUE)
while (length(definition) == 0){
definition <- grep("ENTRY", kegg.reaction((in.kegg)-(scan.row-1))), value=TRUE)
scan.row <- (scan.row + 1)
}
definition <- gsub("ENTRY    ", ",", definition, fixed=TRUE)
definition <- gsub("ENTRY    ", ",", definition, fixed=TRUE)
gene.seq$kegg.reaction.id[i] <- definition

verify if the odonne and paula equations are the same as KEGG

if (as.character(gene.seq$net12.equation[i]) == as.character(gene.seq$kegg.equation[i])){

170
gene.seq$correct <- "YES"
} else {
  gene.seq$correct <- "NO"
}
}
for (g in 2:length(in.kegg)){
  scan.row <- 0
  definition <- grep("sampa", "rio", value=TRUE)
  while (length(definition) == 0){
    definition <- grep("DEFINITION", kegg.reaction[((in.kegg[g])-scan.row)], value=TRUE)
    extra.row <- grep("EQUATION", kegg.reaction[((in.kegg[g])-(scan.row-1))])
    if(length(extra.row) > 0){
      definition <- paste(definition, grep("EQUATION", kegg.reaction[((in.kegg[g])-(scan.row-1))]), fixed=TRUE), sep=" ")
    } else {
      definition <- grep("EQUATION", kegg.reaction[((in.kegg[g])-(scan.row-1))], value=TRUE)
    }
    scan.row <- (scan.row + 1)
  }
  definition <- gsub("DEFINITION ", ",", definition, fixed=TRUE)
  active.row$kegg.equation <- definition
  # get equation with kegg code cpd ######
  scan.row <- 0
  definition <- grep("sampa", "rio", value=TRUE)
  while (length(definition) == 0){
    definition <- grep("EQUATION", kegg.reaction[((in.kegg[g])-scan.row)], value=TRUE)
    scan.row <- (scan.row + 1)
  }
  definition <- gsub("EQUATION ", ",", definition, fixed=TRUE)
  active.row$kegg.equation.codes <- definition
  # get reaction ID ######
  scan.row <- 0
  definition <- grep("sampa", "rio", value=TRUE)
  while (length(definition) == 0){
    definition <- grep("ENTRY", kegg.reaction[((in.kegg[g])-scan.row)], value=TRUE)
    scan.row <- (scan.row + 1)
  }
  definition <- gsub("ENTRY ", ",", definition, fixed=TRUE)
  active.row$kegg.reaction.id <- definition
  # verify if the odonne and paula equations are the same as KEGG ######
  if (as.character(gene.seq$net12.equation[i]) == as.character(active.row$kegg.equation)){
    active.row$correct <- "YES"
  } else {
    active.row$correct <- "NO"
  }
  gene.seq <- rbind(gene.seq, active.row)
}

Appendix 2f.

R script XLStoCSV for Mac OS.

library(tcltk)
library(gdata)
setwd("/Users/ragg005/Documents/MATLAB/model")
excel_file <- "llactismodel09NOV12.xls"
#excel_file <- "llactismodel09NOV12.xls"
sheet1 <- read.xls(excel_file, 1)
names(sheet1)[1] <- "#"

sheet <- "result_script5"
folder <- tk_choose.dir()
store <- paste(folder, "/", sheet, ".csv", sep = "")
write.csv(gene.seq, file = store, row.names = FALSE)
Appendix 2g.

MATLAB function makeSBML for Mac OS.

```matlab
function makeSBML(fileName, outputFileName, version23, printWarnings, ignoreErrors)
%Ignore errors deletes gene info if it encounters any problems. It also %results in that reactions that can't be balanced are not printed (simply %to prevent a lot of text being written)
if nargin<3
    version23=false;
end
if nargin<4
    printWarnings=true;
end
if nargin<5
    %Right now this only deletes all gene information if it finds something %weird there
    ignoreErrors=false;
end
% if nargin<6
%  (fid, sheets)=xlsfinfo(fileName);

%Check if the file is a Microsoft Excel Spreadsheet
if ~strcmp(type, 'Microsoft Excel Spreadsheet')
    throw(MException('', 'The file is not a Microsoft Excel Spreadsheet'));
end
%Check that all sheets are present and saves the index of each
rxnSheet=find(strcmp('RXNS', sheets));
metSheet=find(strcmp('METS', sheets));
compSheet=find(strcmp('COMPS', sheets));
modelSheet=find(strcmp('MODEL', sheets));
geneSheet=find(strcmp('GENES', sheets));
if length(geneSheet)~=1 || length(rxnSheet)~=1 || length(metSheet)~=1 || length(compSheet)~=1
    throw(MException('', 'Not all required spreadsheets are present in the file'));
end
fromR = importdata('MODEL.csv');
discard = fromR.data;
dataSheet = fromR.textdata;
%old version
%[discard, dataSheet]=xlsread(fileName, 'MODEL');

%Find the lines that are not commented
keepers=find(strcmp('', dataSheet(:,1)));
% For the model information there should be only one non-commented line
if length(keepers)==1
% Check if illegal characters are used and abort if that is the case
if ~isempty(dataSheet{keepers,2})
    modelID=dataSheet{keepers,2};
    if ~isempty(regexp(modelID, '^[a-z_A-Z0-9]', 'once'))
        throw(MException('', 'Illegal character(s) in model id'));
    end
    else
        throw(MException('', 'No model ID supplied'));
    end
if ~isempty(dataSheet{keepers,3})
    modelName=dataSheet{keepers,3};
else
    throw(MException('', 'No model name supplied'));
end
% Must check these better!!!
if ~isempty(discard(1))
    defaultLower=discard(1);
end
if ~isempty(discard(2))
    defaultUpper=discard(2);
end
else
    throw(MException('', 'The MODEL sheet should contain only one non-commented line'));
end
% Get compartment information
dataSheet = loadcell('COMPS.csv', ',', '');
dataSheet = regexprep(dataSheet, '""', 'NaN');
dataSheet = regexprep(dataSheet, '" "', 'NaN');
dataSheet = regexprep(dataSheet, '"', 'NaN');
% [discard, discard2, dataSheet] = xlsread(fileName, 'COMPS');
% keepers = findstrcmp('', discard2(:,1));
% Find the lines that are not commented
% keepers = findstrcmp('', discard2(:,1));
% dataSheet = dataSheet(keepers,:);
dataSheet = dataSheet(3:5,:);
% Load compartment info
compAbbrev=dataSheet(:,2);
compName=dataSheet(:,3);
compOutside=dataSheet(:,4);
if version23==true
    compGO=dataSheet(:,5);
else
    compGO={};
end
% Check that both compAbbrev and compName are strings
if ~iscellstr(compAbbrev) || ~iscellstr(compName)
    throw(MException('', 'Both the abbreviated form and compartment name must be strings'));
end
% Check that the abbreviated form only contains one character
% NOTE: I should check to see if it's one of the standard compartments
for i=1:length(compAbbrev)
    if length(compAbbrev{i})~=1 || isempty(regexp(compAbbrev{i}, '[a-zA-Z]', 'once'))
        throw(MException('', 'The abbreviation of compartment ' compName{i} ' does not follow the form of one letter'));
    else
        compAbbrev{i}=lower(compAbbrev{i});
    end
end
% Check to see that all the OUTSIDE compartments are defined
% for i=1:length(compOutside)
% if ~isnan(compOutside{i})
% index=find(strcmp(compOutside{i},compAbbrev),1);
% if isempty(index)
%     throw(MException('','The outside compartment for ' compName{i} ' does not have a corresponding compartment'));
% else
%     compOutside{i}=int2str(index); %Haha, this might be too ugly even for me
% end
% end

for i=1:length(compOutside)
    if ~(strcmp(compOutside(i,:), 'NaN'))
        index=find(strcmp(compOutside(i,:),compAbbrev),1);
        if isempty(index)
            throw(MException('',
                'The outside compartment for ' compName{i} ' does not have a corresponding compartment'));
        else
            compOutside{i}=int2str(index); %Haha, this might be too ugly even for me
        end
    end
end

%Get all the genes and info about them
%[discard, discard2, dataSheet]=xlsread(fileName,'GENES');

dataSheet = loadcell('GENES.csv', ',', '');
dataSheet = regexprep(dataSheet, '\'', '');

%Find the lines that are not commented
keepers=find(strcmp('NA',dataSheet(:,1)));
dataSheet=dataSheet(keepers,:);

%Load the gene information
geneNames=dataSheet(:,2);
geneID1=dataSheet(:,3);
geneID2=dataSheet(:,4);
geneShortNames=dataSheet(:,5);
geneCompartments=dataSheet(:,6);
geneKEGG=dataSheet(:,7);
geneComps={};

%Check that geneName contain only strings and no empty strings
if ~iscellstr(geneNames)
    if ignoreErrors==false
        throw(MException('', 'All gene names have to be strings'));
    else
        deleteGeneInfo();
    end
else
    if ~isempty(find(strcmp('',geneNames),1))
        if ignoreErrors==false
            throw(MException('', 'There can be no empty strings in gene names'));
        else
            deleteGeneInfo();
        end
    end
end

%Check that geneCompart contain only strings and no empty string
if ~iscellstr(geneCompartments)
    if ignoreErrors==false
        throw(MException('', 'All gene compartments have to be strings'));
    else
        deleteGeneInfo();
    end
else
    if ~isempty(find(strcmp('',geneCompartments),1))
        if ignoreErrors==false
            throw(MException('', 'There can be no empty strings in gene compartments'));
        else
            deleteGeneInfo();
        end
    end
end

%Check that all gene compartments correspond to a compartment
for i=1:length(geneNames)
    index=find(strcmp(geneCompartments{i},compAbbrev));
if length(index)==1
geneComps{i}=int2str(index);
else
  if ignoreErrors==false
    throw(MException('','The gene ' geneNames{i} ' has a compartment abbreviation that could not be found'));
  else
    deleteGeneInfo();
  end
end
end

%Check that all gene names are unique
if length(geneNames)~=length(unique(geneNames))
  if ignoreErrors==false
    throw(MException('','Not all gene names are unique'));
  else
    deleteGeneInfo();
  end
end

%Check that geneNames contain no weird characters
illegalCells=regexp(geneNames,'[();:]','once'); %Should check for ';' and ':' too
if ~isempty(cell2mat(illegalCells))
  if ignoreErrors==false
    errorText=['Illegal character(s) in gene names:
    for i=1:length(illegalCells)
      if ~isempty(illegalCells{i})
        errorText=[errorText geneNames{i} '
    end
    throw(MException('','errorText));
  else
    deleteGeneInfo();
  end
end

%New version
data = importdata('RXNS.csv',',');
dataSheet = data.textdata;
dataSheet = dataSheet(2:end,:);
%dataSheet = regexprep(dataSheet,'NA','NaN')
dataSheet(:,end) = num2cell(dataSheet(:,end));
dataSheet = cellfun(@(dataSheet) set2NaN(dataSheet,'NA'), dataSheet, 'UniformOutput', false);
dataSheet = cellfun(@(dataSheet) set2NaN(dataSheet,''), dataSheet, 'UniformOutput', false);

%Loads the reaction data
%[discard,discard2,dataSheet]=xlsread(fileName,'RXNS');
%keepers=find(strcmp('',discard2(:,1))); %Should check for '' too
%databSheet=dataSheet(keepers,:);
reactionIDs=dataSheet(:,2);
reactionNames=dataSheet(:,3);
equations=dataSheet(:,4);
ecNumbers=dataSheet(:,5);
geneAssociations=dataSheet(:,6);

try
  lowerBounds=str2double(dataSheet(:,7));
  upperBounds=str2double(dataSheet(:,8));
  objectives=str2double(dataSheet(:,9));
catch
  throw(MException('','The bounds and objectives must be numerical values'));
end
reactionCompartments=dataSheet(:,10);
reactionSubsystem=dataSheet(:,11);

if version23==true
  reactionReplacement=dataSheet(:,13);
  reactionSBO=dataSheet(:,12);
else
  reactionReplacement={};
  reactionSBO={};
end

%Replace the reaction IDs for those IDs that have a corresponding replacement name. This is only done if version23 is set!
if version23==true
    %The loop is not pretty!
    for i=1:length(reactionIDs)
        if ischar(reactionReplacement{i}) && length(reactionReplacement{i})>0
            reactionIDs(i)=reactionReplacement(i); %This is not ok and has to be fixed at some point
        end
    end
end

%Check that all reaction IDs are unique
if length(reactionIDs)~=length(unique(reactionIDs))
    throw(MException('', 'Not all reaction IDs are unique'));
end

%Checks that the bounds are numerical values
if ~isnumeric(lowerBounds) || ~isnumeric(upperBounds) || ~isnumeric(objectives)
    throw(MException('', 'The bounds and objectives must be numerical values'));
end

%Check that there are no empty strings in reactionIDs or equations
if iscellstr(reactionIDs) && iscellstr(equations)
    if ~isempty(find(strcmp('',reactionIDs),1)) || ~isempty(find(strcmp('',equations),1))
        throw(MException('', 'There are empty reaction IDs or equations'));
    else
        throw(MException('', 'Both reaction IDs and equations must be strings only'));
    end
else
    throw(MException('', 'reactionIDs or equations must be strings only'));
end

%Check that reactionIDs contain no weird characters
illegalCells=regexp(reactionIDs, '[^a-zA-Z0-9]', 'once');
if ~isempty(cell2mat(illegalCells))
    errorText=['Illegal character(s) in reaction IDs:
    for i=1:length(illegalCells)
        errorText=[errorText reactionIDs{i} ' 
    end
    throw(MException('', errorText));
end

%Check that all reactions have compartments defined
if ~iscellstr(reactionCompartments)
    throw(MException('', 'All reactions must have an associated compartment string'));
else
    if ~isempty(find(strcmp('',reactionCompartments),1))
        throw(MException('', 'All reactions must have an associated compartment string'));
end
end

%Check that an SBO-term is associated with each reaction. This is only
%done if version23=true
if version23==true
    if ~iscellstr(reactionSBO)
        reactionSBO=[];
    else
        if ~isempty(find(strcmp('',reactionSBO),1))
            reactionSBO=[];
    end
end
end

%Used if gene associations should be deleted if problems
foundGeneProblem=false;

%Replace empty reaction names with the corresponding reaction ID and
%calculate the compartment index
for i=1:length(reactionNames)
    if isempty(reactionNames{i}) || isstr('reactionNames{i}''=''; %I think this is more correct
    end

%Check that all gene associations have a match in the gene list
if ischar(geneAssociations{i}) && length(geneAssociations{i})>0
    indexes=strfind(geneAssociations{i},':'); %Genes are separated by ":" for AND and ";" for OR
    indexes=unique([indexes strfind(geneAssociations{i},'\''']));
if isempty(indexes)
    % See if you have a match (it can't have more than one since the
    % names are unique)
    if isempty(find(strcmp(geneAssociations{i},geneNames),1))
        if ignoreErrors==false
            throw(MException('',["The gene association in reaction 'reactionIDs{i}'
                                 ('geneAssociations{i}') is not present in the gene list']);
        end
        foundGeneProblem=true;
    else
        temp=[0 indexes numel(geneAssociations{i})+1];
        for j=1:numel(indexes)+1;
            % The reaction has several associated genes
            geneName=geneAssociations{i}(temp(j)+1:temp(j+1)-1);
            if isempty(find(strcmp(geneName,geneNames),1))
                if ignoreErrors==false
                    throw(MException('',["The gene association in reaction 'reactionIDs{i}'
                                            ('geneName') is not present in the gene list']);
                else
                    foundGeneProblem=true;
                end
            end
        end
    end
end

% Check that the compartment for each reaction can be found and save the
% position in compAbbrev
index=find(strcmp(reactionCompartments{i},compAbbrev));
if length(index)==1
    reactionComps{i}=int2str(index);
else
    throw(MException('',["The reaction 'reactionNames{i}' has a compartment abbreviation
                         that could not be found']);)
end
end

% Get all the metabolites and info about them
% [discard, discard2, dataSheet]=xlsread(fileName,'METS');
% data = loadcell('METS', '\tab', '')
dataSheet = loadcell('METS', [',', char(9)], '', 'single-string');
dataSheet = dataSheet(2:end, :);
dataSheet = cellfun(@(dataSheet) set2NaN(dataSheet, 'NA'), dataSheet, 'UniformOutput', false);
dataSheet = cellfun(@(dataSheet) regexprepcarchar(dataSheet, 'ab', ''), dataSheet, 'UniformOutput', false);
dataSheet(:,9) = regexprepcarchar(dataSheet(:,9), 'ab', '');
dataSheet = cellfun(@(dataSheet) set2NaN(dataSheet, ''), dataSheet, 'UniformOutput', false);
% dataSheet(:,1) = sum(nans, 1)
% keepers=find(strncmp('', discard2(:,1)));
% dataSheet=dataSheet(keepers,:);

% Load the metabolite information
metaboliteIDs=dataSheet(:,2);
metaboliteNames=dataSheet(:,3);
metConstrained=dataSheet(:,4);
metMiriam=dataSheet(:,5);
metComposition=dataSheet(:,6);
metInchi=dataSheet(:,13); %%% ORIGINALLY 7
metCompartments=dataSheet(:,14); %%% Originally 8
metReplacement=dataSheet(:,15); %%% Originally 9

%finalMetIDs=metaboliteIDs;

%Replace the metabolite IDs for those IDs that have a corresponding replacement metabolite. This is not used for matching, but will be checked for consistency with SBML naming conventions

%The loop is not pretty!
for i=1:length(metaboliteIDs)
    if ~isempty(metReplacement{i}) && ~all(isnan(metReplacement{i}))
        finalMetIDs(i,1)=metReplacement(i);
    end
end

metaboliteIDs = metaboliteIDs(1:length(finalMetIDs),1);
metaboliteNames = metaboliteNames(1:length(finalMetIDs),1);
metConstrained = metConstrained(1:length(finalMetIDs),1);
metMiriam = metMiriam(1:length(finalMetIDs),1);
metComposition = metComposition(1:length(finalMetIDs),1);
metInchi = metInchi(1:length(finalMetIDs),1);
metCompartments = metCompartments(1:length(finalMetIDs),1);
metReplacement = metReplacement(1:length(finalMetIDs),1);

%Replace all metabolite ids with replacement IDs (if available)
lowerMetaboliteIDs=lower(metaboliteIDs);

%Check that metaboliteIDs contain only strings, no empty strings and no weird characters.
%finalMetIDs = cell2mat(finalMetIDs);
%finalMetIDs = num2str(finalMetIDs);
%finalMetIDs = cellstr(finalMetIDs);
finalMetIDs = regexprep(finalMetIDs, ' ', '');

if ~iscellstr(finalMetIDs)
    throw(MException('','All metabolite IDs have to be strings'));
else
    if ~isempty(find(strcmp('',finalMetIDs),1))
        throw(MException('','There can be no empty strings in metabolite IDs'));
    end
end

%Check that all metabolites have compartments defined
if ~iscellstr(metCompartments)
    throw(MException('','All metabolites must have an associated compartment string'));
else
    if ~isempty(find(strcmp('',metCompartments),1))
        throw(MException('','All metabolites must have an associated compartment string'));
    end
end

for i=1:length(finalMetIDs)
    %Check that the compartment for each metabolite can be found and save to %POSITION IN COMPNAME!!!
    index=find(strcmp(metCompartments{i},compAbbrev));
    if length(index)==1
        metComps{i}=index;
    else
        throw(MException('','The metabolite ' finalMetIDs{i} ' has a compartment abbreviation that could not be found'));
    end
end

%Check that the "constrained" fields are "true", "false", or NaN
if iscellstr(metConstrained(i))
    if isempty(find(strcmp('false',lower(metConstrained(i)))),1)) &&
        isempty(find(strcmp('true',lower(metConstrained(i)))),1))
        throw(MException('','The UNCONSTRAINED property for metabolite ' finalMetIDs{i} ' must be "true", "false", or not set'));
    else
        metConstrained{i}=lower(metConstrained{i});
    end
else
    if ~isnan(metConstrained{i})
        throw(MException('','The UNCONSTRAINED property for metabolite ' finalMetIDs{i} ' must be "true", "false", or not set'));
    else
        metConstrained{i}='false';
    end
end
%If the metabolite name isn't set, replace it with the metabolite id
if ~ischar(metaboliteNames{i}) || isempty(metaboliteNames{i})
    metaboliteNames(i)=metaboliteIDs(i);
end
%Check that it doesn't contain any forbidden characters
illegalCells=regexp(metaboliteNames,'["<>\"\']','once');
if ~isempty(cell2mat(illegalCells))
    errorText=['Illegal character(s) in metabolite names: 
    for i=1:length(illegalCells)
        if ~isempty(illegalCells{1})
            errorText=[errorText metaboliteNames{i} ' 
        end
    end
    throw(MException('',errorText));
end
%Everything seems fine with the metabolite IDs, compartments, genes, and reactions
revIndexes=strfind(equations,' <=> ');
irrevIndexes=strfind(equations,' => ');
if any(cellfun(@isempty,revIndexes) & cellfun(@isempty,irrevIndexes))
    throw(MException('','The reaction ' reactionIDs{find(cellfun(@isempty,revIndexes) & cellfun(@isempty,irrevIndexes),1)} ' does not have reversibility data'));
end
%Split the reactions in left and right side
lhs=cell(numel(equations),1);
rhs=cell(numel(equations),1);
for i=1:numel(equations)
    stop=[revIndexes{i} irrevIndexes{i}];
    lhs{i}=equations{i}(1:stop(1)-1);
    if isempty(revIndexes{i})
        rhs{i}=equations{i}(stop(1)+4:end);
    else
        rhs{i}=equations{i}(stop(1)+5:end);
    end
end
%This code is extremely badly written, but it's all very temporary. Must be fixed!
leftPlus=strfind(lhs,' + ');
rightPlus=strfind(rhs,' + ');
leftSpace=strfind(lhs,' ');
rightSpace=strfind(rhs,' ');
%Preallocate a METxRNS stoichiometric matrix
S=sparse(zeros(length(metaboliteIDs),length(reactionIDs)));
%Loop through each of the equations
for i=1:numel(equations)
    for k=1:2:1  %-1 is for reactant side
        if k==1
            currentPlus=leftPlus;
currentSpace=leftSpace;
currentSide=lhs;
else
            currentPlus=rightPlus;
currentSpace=rightSpace;
currentSide=rhs;
end
currentPlus=rightPlus;
currentSpace=rightSpace;
currentSide=rhs;
end

starts=[1 currentPlus{i}+3 numel(currentSide{i})+4];
for j=1:numel(starts)-1
  % Check to see if it starts with a coefficient
  coeff=str2double(currentSide{i}(starts(j):currentSpace{i}(find(currentSpace{i}>starts(j),1))-1));
  if ~isnan(coeff)
    % If it starts with a coefficient
    metName=currentSide{i}(currentSpace{i}(find(currentSpace{i}>starts(j),1))+1:starts(j+1)-4);
  else
    coeff=1;
    metName=currentSide{i}(starts(j):starts(j+1)-4);
  end
  % Check to see that it was found in the list
  % Check if the metabolite is present
  metID=find(strcmp(lower(metName),lowerMetaboliteIDs),1);
  % If it didn't find the metabolite
  if isempty(metID)
    throw(MException('',['The metabolite ' metName ' in reaction ' reactionIDs{i}
      ' was not found in the metabolite list']));
  end
  % If the metabolite was present in more than one copy
  if length(metID)>1
    throw(MException('',['The metabolite ' metName ' in reaction ' reactionIDs{i}
      ' was found in more than one copy in the metabolite list']));
  end
  % If the metabolites don't match cases
  if strcmp(metName,metaboliteIDs(metID))~=1
    fprintf('WARNING: The metabolite %s in reaction %s differs in upper/lower case
      compared to the metabolite list\n',metName,reactionIDs{i});
  end
  % Check to see that the metabolite isn't already present in
  % the reaction. This means that the reaction is on the form
  % A => A
  if printWarnings==true
    if S(metID,i)==0
      fprintf(['WARNING: The reaction ' reactionIDs{i} ' has one or more
        metabolites both as reactants and as products. Only the net reaction will be exported\n']);
    end
  end
  S(metID,i)=S(metID,i)+coeff*k;
end
end

reversibility=zeros(1,length(equations));
reversibility(~cellfun(@isempty,revIndexes))=1;

if printWarnings==true
  % Check that all the metabolites are being used
  involvedMat=S;
  involvedMat(find(abs(involvedMat)>0))=1;
  usage=sum(involvedMat');
  notPresent=find(usage==0);
  unbalanced=find(usage==1);
  if ~isempty(notPresent)
    errorText=['WARNING: The following metabolite(s) are never used:\n'];
    for i=1:length(notPresent)
      errorText=errorText ['(' finalMetIDs{notPresent(i)} ') ' metaboliteNames{notPresent(i)} '\n'];
    end
    fprintf([errorText '\n']);
  end
end
%Note: This should take reactants/products into account. If a
%metabolite is only a product in all (irreversible) reactions, then is
%is it unbalanced?
if ~isempty(unbalanced)
  errorText=['WARNING: The following internal metabolite(s) are only used in one
reaction (zero flux is the only solution):

for i=1:length(unbalanced)
  %Assumes the same naming that I have. A little problematic
  if ~strcmp(compAbbrev(metComps{unbalanced(i)}),'b') &&
  ~strcmp(compAbbrev(metComps{unbalanced(i)}),'e')
    errorText=[errorText ' (' finalMetIDs{unbalanced(i)} ' ['
    compAbbrev(metComps{unbalanced(i)})) ' ]) metaboliteNames{unbalanced(i)} '

end
end
fprintf(errorText ']

end

%Check that all metabolites are balanced for C, N, S, P and the number
%of R-groups
[nC nN nS nP foundComp]=getComposition(metComposition, metInchi);

%Loop through the reactions and look at those where all the metabolites
%have composition data. Just count the others
cantbalanceRxns=[];
for i=1:size(S,2)
  foundMets=find(S(:,i));
  if sum(foundComp(foundMets))==numel(foundMets)
    cBalance=sum(S(foundMets,i).*nC(foundMets));
    nBalance=sum(S(foundMets,i).*nN(foundMets));
    sBalance=sum(S(foundMets,i).*nS(foundMets));
    pBalance=sum(S(foundMets,i).*nP(foundMets));
    %Arbitrarily small number
    if abs(cBalance(1,1))>0.00000001
      fprintf('WARNING: The reaction %s is not balanced with respect to
carbon
',reactionIDs{i});
    end
    if abs(nBalance(1,1))>0.00000001
      fprintf('WARNING: The reaction %s is not balanced with respect to
nitrogen
',reactionIDs{i});
    end
    if abs(sBalance(1,1))>0.00000001
      fprintf('WARNING: The reaction %s is not balanced with respect to
sulfur
',reactionIDs{i});
    end
    if abs(pBalance(1,1))>0.00000001
      fprintf('WARNING: The reaction %s is not balanced with respect to
phosphorus
',reactionIDs{i});
    end
    else
      cantbalanceRxns=[cantbalanceRxns i];
    end
  end
if ignoreErrors==false
  fprintf('
WARNING: The following %s reactions could not be checked for mass
balancing
',num2str(numel(cantbalanceRxns)));
  fprintf('(%s
',reactionIDs(cantbalanceRxns));
else
  fprintf('WARNING: %s reactions could not be checked for mass
balancing
',num2str(numel(cantbalanceRxns)));
end
%Check that an SBO-term is associated with each reaction. This is only
%done if version23=true
if version23==true
  if isempty(reactionSBO)
    fprintf('WARNING: Not all reactions have associated SBO-terms. SBO-terms will
not be used.
');
  end
end

%All the information has been collected. Time to write SBML!
exportSBML23(outputFileName,modelID,modelName,finalMetIDs,metaboliteNames,...
metMiriam, metComposition, metInchi, metConstrained, metComps, reactionIDs, reactionNames,
reactionComps, S, reversibility,...
%This is a temporary thing that works on non-local variables in a very
%unneat way. Delete this whole thing when you have fixed everything with ignoreErrors
function deleteGeneInfo()
geneNames={};
geneID1={};
geneID2={};
geneShortNames={};
geneCompartments={};
geneKEGG={};
geneComps={};
if exist('geneAssociations', 'var')
geneAssociations(:)={NaN};
end
%Print a warning
fprintf('WARNING: An error was found regarding genes. All gene related information has
been deleted. Set ignoreErrors to false to get more specific information about the
error.\n');
end
end
function [C N S P foundComp]=getComposition(metComposition, metInchi)%Assume that they are of the same length
C=zeros(size(metComposition));
N=zeros(size(metComposition));
S=zeros(size(metComposition));
P=zeros(size(metComposition));
foundComp=zeros(size(metComposition));
for i=1:length(metComposition)
    inchiError=0;
    if ischar(metInchi{i})
        if length(metInchi{i})>0
            %Find the formula in the Inchi string. Assume that it is
            %everything between the first and second "/" index;
            indexes=strfind(metInchi{i}, '/');
            if length(indexes)==0
                inchiError=1;
            else
                %For some simple molecules such as salts only the first "/" is present
                if length(indexes)==1
                    formula=metInchi{i}(indexes(1)+1:length(metInchi{i}));
                else
                    formula=metInchi{i}(indexes(1)+1:indexes(2)-1);
                end
                [nC, nN, nS, nP, errorFlag] = compFromFormula(formula);
                if errorFlag==0
                    C(i)=nC;
                    N(i)=nN;
                    S(i)=nS;
                    P(i)=nP;
                    foundComp(i)=1;
                end
            end
        else
            inchiError=1;
        end
    else
        inchiError=1;
    end
    if inchiError==1
        %If no InChI could be found
        if ischar(metComposition{i})
            if length(metComposition{i})>0
                [nC, nN, nS, nP, errorFlag] = compFromFormula(metComposition{i});
                if errorFlag==0
                    C(i)=nC;
                end
            end
        end
    end
end
183

N(i)=nN;
S(i)=nS;
P(i)=nP;
foundComp(i)=1;
end
end
end
end
end
end
end
end
end

function [nC, nN, nS, nP, errorFlag] = compFromFormula(formula)
%IMPORTANT! This does not work if elements can have more than one
%character. Look into that!
errorFlag=0;
nonNumeric=regexp(formula,'[^0-9]');
abbrevs=['C';'N';'S';'P'];
comp=zeros(size(abbrevs));
for i=1:size(abbrevs)
index=strfind(formula,abbrevs(i));
if length(index)==1
nextNN=find(nonNumeric>index);
if ~isempty(nextNN)
comp(i)=str2double(formula(index+1:nonNumeric(nextNN(1))-1));
else
comp(i)=str2double(formula(index+1:length(formula)));
end
%Might be temporary!! Assumes that there is 1 atom if the
%str2double thing didn't work
if isnan(comp(i))
comp(i)=1;
end
else
if length(index)>1
%THIS HAS TO BE FIXED! THIS CAN HAPPEN FOR POLYMERS FOR
%EXAMPLE!!
errorFlag=1;
end
end
end
nC=comp(1);
nN=comp(2);
nS=comp(3);
P=comp(4);
end

function [errorFlag] = exportSBML23(outputFile,modelID,modelName,metaboliteIDs,...
metaboliteNames, metMiriam, metComposition, metInchi, metConstrained,
metComps,reactionIDs, reactionNames,...
reactionComps, stochiometricMatrix, reversibility, compName, compOutside, compAbbrev,
compGO, lowerBounds,...
upperBounds, objectives, ecNumbers, geneAssociations, reactionSubsystem, reactionSBO,
defaultLower,defaultUpper,...
geneNames, geneID1, geneID2, geneShortNames, geneComps, geneKEGG, version23)

%Generate temporary name
tempFile=tempname;
if version23==false
metaboliteNames=strrep(metaboliteNames, ' ', '=');
metaboliteNames=strrep(metaboliteNames, '-', '=');
end
%Construct a Model file
model.rxns=reactionIDs;
model.mets=metaboliteIDs;
model.S=stochiometricMatrix;
model.rev=reversibility;
formulas=printRxnFormula(model,reactionIDs,false);
 fid = fopen(tempFile, 'w');
if version23==true
intro=['<?xml version="1.0" encoding="UTF-8" ?>''
'<model metaid="metaid_ ' modelID ' id="' modelID ' " name="' modelName '"'>''
'<notes>'...
'<body xmlns="http://www.w3.org/1999/xhtml">''

183
This is a reconstruction of the biochemical network of the yeast *Saccharomyces cerevisiae*. There are no kinetic functions defined for the reactions because this model only represents the chemical structure of the network (stoichiometry). All uptake reactions are constrained to zero flux and all excretion reactions are unconstrained. All genes are assigned to the cytosol. This has no physiological meaning, but it is necessary for the structure of the model. Technical notes:

- The compartments included here have no volume defined; there are no reliable estimates available for those volumes yet.
- There are no kinetic functions defined for the reactions because this model only represents the chemical structure of the network (stoichiometry).
- Uptake and excretion are defined for some of the metabolites. All uptake reactions are constrained to zero flux and all excretion reactions are unconstrained.
- All genes are assigned to the cytosol. This has no physiological meaning, but it is necessary for the structure of the model.
- Biomass equations are available for carbon-limited and nitrogen-limited growth. The nitrogen-limited biomass equation is constrained to zero flux.
- A few reactions are meant to be used to simulate the effect of for example increased ATP production. They are constrained to zero flux.

This SBML representation of the *Saccharomyces cerevisiae* metabolic network is made available under the Creative Commons Attribution-Share Alike 3.0 Unported Licence (see <a href="http://www.creativecommons.org"">www.creativecommons.org</a>).
This is a reconstruction of the biochemical network of the yeast <i>Saccharomyces cerevisiae</i>.<p>

Technical notes:<ul><li>The compartments included here have no volume defined; there are no reliable estimates available for those volumes yet.</li><li>There are no kinetic functions defined for the reactions because this model only represents the chemical structure of the network (stoichiometry).</li><li>Reactions for uptake and excretion are defined for some of the metabolites. All uptake reactions are constrained to zero flux and all excretion reactions are unconstrained.</li></ul>This SBML representation of the yeast metabolic network is made available under the Creative Commons Attribution-Share Alike 3.0 Unported Licence (see <a href="http://www.creativecommons.org">www.creativecommons.org</a>).
toprint=['<species metaid="metaid_M_' metaboliteIDs[i] '" id="M_' metaboliteIDs[i] '" name="' metaboliteNames[i] '" compartment="C_' int2str(metComps[i]) '" boundaryCondition="" metConstrained[i] '" sboTerm="SBO:0000299";'>
else
  %Get the formula for the compound
  if ischar(metComposition[i]) && length(metComposition[i])>0
    if ~isempty(regexp(metComposition[i],'[a-zA-Z0-9]', 'once'))
      formula=''_';
    else
      formula=''_ ' metComposition[i];
    end
  else
    %Find the formula in the Inchi string. Assume that it is
    everything between the first and second "\n" indexes=strfind(metInchi[i],'/');
    if length(indexes)<2
      formula=''_';
    else
      if ~isempty(regexp(metInchi[i](indexes(1)+1:indexes(2)-1),'[a-zA-Z0-9]',
        'once'))
        formula=''_';
      else
        formula=''_ ' metInchi[i](indexes(1)+1:indexes(2)-1);)
      end
    end
  end
  %style: ';%TEMP, DELETE. This is only for Robert1
  toprint=['<species id="M_' metaboliteIDs[i] '_compAbbrev{metComps[i]}" name="' metaboliteNames[i] formula
    '' compartment="C_' int2str(metComps[i]) '" bound
    aryCondition="" metConstrained[i] '"">'];
end
%Print some stuff if there is a formula for the compound
if version23==true
  if ischar(metComposition[i])
    if length(metComposition[i])>0
      toprint=[toprint '<notes><
        html xmlns="http://www.w3.org/1999/xhtml"><p>FORMULA: ' metComposition[i] '
        </p></html></notes>'
    end
  end
end
%Only print annotations for metabolites with some miriam link. This is because I don't
know how "unknown" metabolites should be presented, and it seems unlikely that you will
have InChI without a database link. This might be temporary....
if version23==true
  if ischar(metMiriam[i]) && length(metMiriam[i])>0
    toprint=[toprint '<annotation>
      %Print InChI if available
      if ischar(metInchi[i])
        if length(metInchi[i])>0
          toprint=[toprint '<in:inchi xmlns:in="http://biomodels.net/inchi"
            metaid="metaid_M_' metaboliteIDs[i] '_inchi">InChI="' metInchi[i] '
          </in:inchi>'
        else
          isInchi=1;
        end
      else
        isInchi=0;
      end
    else
      isInchi=0;
    end
  %Print some more annotation stuff
  toprint=[toprint '<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:dcterm="http://purl.org/dc/terms/" ...
    xmlns:vCard="http://www.w3.org/2001/vcard-rdf/3.0#"
    xmlns:bqbiol="http://biomodels.net/biology-qualifiers/" ...
    xmlns:bqmodel="http://biomodels.net/model-qualifiers/"">
    <rdf:Description rdf:about="#metaid_M_' metaboliteIDs[i] '
      <in:inchi xmlns:in="http://biomodels.net/inchi"
        metaid="metaid_M_' metaboliteIDs[i] '_inchi"InChI="' metInchi[i] '
      </in:inchi>'
    ...
    <bqbiol:is>
      ...
      <rdf:li rdf:resource="#metaid_M_' metaboliteIDs[i] '_inchi" />
    </bqbiol:is>
  </rdf:RDF>]
end
%Print miriam
toprint=[toprint '<rdf:li rdf:resource="urn:miriam:_' metMiriam[i] '
    "'/>'];
%Finish up
%Add information on all modifiers (that is the genes)
%Loop through to replace empty cells with '', NOT PRETTY!
for i=1:length(geneAssociations)
    geneAssociations(i)='';
end
if version23==true
%First add all the genes
    for i=1:length(geneNames)
        geneNames(i)='';
        if ischar(geneShortNames(i)) && length(geneShortNames(i))>0
            fprintf(fid,toprint);
        end
        %Print annotation info if present
        if (ischar(geneKEGG{i}) && length(geneKEGG{i})>1) || (ischar(geneID1{i}) && length(geneID1{i})>1) || (ischar(geneID2{i}) && length(geneID2{i})>1)
            fprintf(fid,toprint);
        end
        [crap crap crap crap crap crap keggMaps] = regexp(geneKEGG{i}, '[:]+');
        for j=1:length(keggMaps)
            fprintf(fid,toprint);
        end
        %Print gene name if present
        if ischar(geneID1{i}) && length(geneID1{i})>0
            fprintf(fid,toprint);
        end
        if ischar(geneID2{i}) && length(geneID2{i})>0
            fprintf(fid,toprint);
        end
        if ischar(geneKEGG{i}) && length(geneKEGG{i})>0
            fprintf(fid,toprint);
        end
        fprintf(fid,toprint);
    end
end
%Loop through all reactions and find gene associations which contain
%'', which means that they are governed by several genes
uniqueGenes=unique(geneAssociations);
[crap reactions crap crap crap crap crap keggMaps] = regexp(uniqueGenes, ''[:]+''');
reactions=find(cellfun(''length'', reactions));
[crap crap crap crap crap mods] = regexp(uniqueGenes(reactions), ''[:]+''');
geneComplexes={};
complexGenes={};
%Loop through each modifier and add the ones that are complexes
for i=1:numel(reactions)
    %Check to see if it's a complex
    [crap complex crap crap crap crap comGenes] = regexp(mods{i}, ''[:]+''');
    complex=find(cellfun(''length'', complex));
    for j=1:numel(complex)
        geneComplexes{geneComplexes{mods{i}(complex(j))};
end
complexGenes=[complexGenes;comGenes(complex(j))];
end

%Remove duplicate complexes
[geneComplexes,I]=unique(geneComplexes);

%The SBO term for the complex is set to be "protein". Might not be correct.
for i=1:length(geneComplexes)
    toprint=['<species metaid="metaid_Cx_ ' int2str(i) '" id="Cx_ ' int2str(i) '" name="
geneComplexes(i) '" compartment="C_2" sboTerm="SBO:0000297">'];
    toprint=['</species>
    fprintf(fid,toprint);
end

%Finish metabolites
fprintf(fid, '</listOfSpecies>

%Add reactions
fprintf(fid, '<listOfReactions>
for i=1:length(reactionIDs)
%Get reversibility
reversible='false';
if reversibility(i)==1
    reversible='true';
end
if version23==true
    if ~isempty(reactionSBO)
        SBO=[' sboTerm="' reactionSBO(i) '"'];
    else
        SBO='';
    end
    fprintf(fid, ['<reaction metaid="metaid_R_ ' reactionIDs(i) '" id="R_ ' reactionIDs(i) '" name="' reactionNames(i) '" reversible="' reversible '" SBO='']);
    else
        fprintf(fid, ['<reaction id="R_ ' reactionIDs(i) '" name="' reactionNames(i) '" reversible="' reversible '"
        if version23=true
            if ischar(reactionSubsystem(i)) && length(reactionSubsystem(i))>0
                fprintf(fid,'<notes>');
            end
        else
            if ~isnan(geneAssociations(i))
                geneString=strrep(geneAssociations(i),':',' and ');
                geneString=strrep(geneString,';',' or ');
                fprintf(fid, ['<html xmlns="http://www.w3.org/1999/xhtml"><p>GENE_ASSOCIATION: (' geneString ')</html:p>']);
            end
        end
    end

if version23=true
    if ischar(ecNumbers(i)) && length(ecNumbers(i))>0
fprintf(fid,'</listOfReactants>');</nml:rdf>
%The reactants have negative values in the stochiometric matrix
compounds=stochiometricMatrix(:,i);
reactants=find(compounds<0);
products=find(compounds>0);
for j=1:length(reactants)
    if version23==true
        tempmetname=metaboliteIDs{reactants(j)};
    else
        tempmetname=[metaboliteIDs{reactants(j)} '_' compAbbrev{metComps{reactants(j)}}];
    end
    toprint=['<speciesReference species="M_' tempmetname '" stoichiometry="'
        char(num2str(-1*compounds(reactants(j)))) '"/>'];
    fprintf(fid,toprint);
end
fprintf(fid,'</listOfReactants><listOfProducts>');</nml:rdf>
for j=1:length(products)
    if version23==true
        tempmetname=metaboliteIDs{products(j)};
    else
        tempmetname=[metaboliteIDs{products(j)} '_' compAbbrev{metComps{products(j)}}];
    end
    toprint=['<speciesReference species="M_' tempmetname '" stoichiometry="'
        char(num2str(compounds(products(j)))) '"'/>'];
    fprintf(fid,toprint);
end
fprintf(fid,'</listOfProducts>');</nml:rdf>
if version23==true
    %Print modifiers if available.
    if ischar(geneAssociations{i}) && length(geneAssociations{i})>0
        %Loop through the number of modifiers (isoenzymes, complexes...)
        [crap crap crap crap crap crap mods]=regexp(geneAssociations{i},'[;]');
        fprintf(fid,'<listOfModifiers>');
        for j=1:numel(mods)
            if isempty(strfind(mods{j},';'))
                %Find the correct gene
                index=find(strcmp(mods{1},geneNames),1);
                %Assumes that it is found since that check should have been made
                fprintf(fid,'<modifierSpeciesReference species="E_%s"/>',num2str(index(1)));
            else
                index=find(strcmp(mods{j},geneComplexes),1);
                %Assumes that it is found since that check should have been made
                fprintf(fid,'<modifierSpeciesReference species="Cx_%s"/>',num2str(index(1)));
            end
        end
        fprintf(fid,'</listOfModifiers>');
end
%Print constraints
if isnan(upperBounds(i))
    upper=defaultUpper;
else
    upper=upperBounds(i);
end
if isnan(lowerBounds(i))
    %Check for reversibility
    if reversibility(i)==1
        lower=defaultLower;
    else
        lower=0;
end

fprintf(fid,'</nml:rdf>');
else
    lower=lowerBounds(i);
end

%Print objectives
if isnan(objectives(i))
    objective=0;
else
    objective=objectives(i);
end

if version23==true
    fprintf(fid,'<kineticLaw><math
    xmlns="http://www.w3.org/1998/Math/MathML"><ci>FLUX_VALUE</ci></math><listOfParameters>
    <parameter id="LB_R_ reactionIDs{i}" name="LOWER_BOUND" value="sprintf('%15.8f',lower)"
    units="mmol_per_gDW_per_hr"/>
    <parameter id="UB_R_ reactionIDs{i}" name="UPPER_BOUND" value="sprintf('%15.8f',upper)"
    units="mmol_per_gDW_per_hr"/>
    <parameter id="OBJ_R_ reactionIDs{i}" name="OBJECTIVE_COEFFICIENT" value="sprintf('%15.8f',objective)"
    units="dimensionless"/>
    <parameter id="FLUX_VALUE" value="0.00000000" units="mmol_per_gDW_per_hr"/>
    </listOfParameters></kineticLaw>');
else
    fprintf(fid,'<kineticLaw><math xmlns="http://www.w3.org/1998/Math/MathML"><apply><ci>
    LOWER_BOUND </ci><ci> UPPER_BOUND </ci><ci> OBJECTIVE_COEFFICIENT </ci></apply><listOfParameters>
    <parameter id="LOWER_BOUND" value="sprintf('%15.8f',lower)"
    units="dimensionless"/>
    <parameter id="UPPER_BOUND" value="sprintf('%15.8f',upper)"
    units="mmol_per_gDW_per_hr"/>
    <parameter id="OBJECTIVE_COEFFICIENT" value="sprintf('%15.8f',objective)"
    units="dimensionless"/>
    </listOfParameters></kineticLaw>');
end

fprintf(fid,'</listOfParameters><reaction>

%Add reactions for the creation of complexes
if version23==true
    for i=1:length(geneComplexes)
        fprintf(fid,'<reaction metaid="metaid_R_ int2str(length(reactionIDs)+i)"
            id="R_ int2str(length(reactionIDs)+i)"
            name="strrep(geneComplexes{i},':','-',')"
            reversible="false" sbTerm="SBO:0000176">
            <listOfReactants>
            <speciesReference species="E_int2str(find(strcmp(complexGenes{i,1}{j},geneNames)))"
                stoichiometry="1"></speciesReference>
            <speciesReference species="Cx_i"
                stoichiometry="1"></speciesReference>
            </listOfReactants>
            <kineticLaw><math
                xmlns="http://www.w3.org/1998/Math/MathML"><ci>FLUX_VALUE</ci></math><listOfParameters>
                <parameter id="LB_R_ int2str(length(reactionIDs)+i)"
                    name="LOWER_BOUND" value="0.00000000" units="mmol_per_gDW_per_hr"/>
                <parameter id="UB_R_ int2str(length(reactionIDs)+i)"
                    name="UPPER_BOUND" value="defaultUpper"
                    units="mmol_per_gDW_per_hr"/>
                <parameter id="OBJ_R_ int2str(length(reactionIDs)+i)"
                    name="OBJECTIVE_COEFFICIENT" value="0.00000000" units="dimensionless"/>
                <parameter id="FLUX_VALUE" value="0.00000000" units="mmol_per_gDW_per_hr"/>
                </listOfParameters></kineticLaw>
            </reaction>>
        end
    end
else
    fprintf(fid,'<listOfReactions>'
    fprintf(fid,'</listOfReactions>'

%Write outro
fprintf(fid,'</model></sbml>'
fclose(fid);
end

%Replace the target file with the temporary file
delete(outputFile);
movefile(tempFile,outputFile);
errorFlag=0;
function formulas = printRxnFormula(model,rxnNameList,printFlag,lineChangeFlag,fid)
%printRxnFormula Print the reaction formulas for a list of reactions
% formulas = printRxnFormula(model,rxnNameList,printFlag,lineChangeFlag,fid)
%    model             Constraint-
%                      based model
%    rxnNameList      Names of reactions whose formulas are to be printed
%    printFlag        Print formulas or just return them (opt, default true)
%    lineChangeFlag   Append a line change at the end of each line (opt, default true)
%    fid              Optional file identifier for printing in files

%ALL TEMPORARY!!!!!!!!!!!!!!!
if (nargin < 2)
    rxnNameList = model.rxns;
end
if (nargin < 3)
    printFlag = true;
end
if (nargin < 4)
    lineChangeFlag = true;
end
if (nargin < 5)
    fid = 1;
end
if (~iscell(rxnNameList))
    if (strcmp(rxnNameList,'all'))
        rxnNameList = model.rxns;
    else
        rxnNameTmp = rxnNameList;
        clear rxnNameList;
        rxnNameList{1} = rxnNameTmp;
    end
end
for i = 1:length(rxnNameList);
    rxnName = rxnNameList{i};
    rxnID = findRxnIDs(model,rxnName);
    if (printFlag)
        fprintf(fid,'%s'
        fprintf(fid,'\t',rxnName);
    end
    if (rxnID > 0)
        Srxn = model.S(:,rxnID);
        if (isfield(model,'ub') & model.ub(rxnID) <= 0)
            Srxn = -Srxn;
        end
        Sprod = Srxn(Srxn > 0);
        prodMets = model.mets(Srxn > 0);
        Sreact = Srxn(Srxn < 0);
        reactMets = model.mets(Srxn < 0);
        formulaStr = ''; 
        for j = 1:length(reactMets)
            if (j > 1)
                if (printFlag)
                    fprintf(fid,'+ ');
                end
                formulaStr = [formulaStr '+ '];
            end
            if (abs(Sreact(j)) ~= 1)
                if (printFlag)
                    fprintf(fid,'%f %s ',abs(Sreact(j)),reactMets{j});
                end
                formulaStr = [formulaStr num2str(abs(Sreact(j))) ' ' reactMets{j} ' '];
            else
                if (printFlag)
                    fprintf(fid,'%s ',reactMets{j});
                end
            end
        end
    end
end
end
formulaStr = [formulaStr reactMets{j} ' '];
end

if (model.rev(rxnID))
if (printFlag)
fprintf(fid,'\t&lt;-&gt;\t');
end
formulaStr = [formulaStr ' &lt;-&gt; '];
else
if (printFlag)
fprintf(fid,'\t-&gt;\t');
end
formulaStr = [formulaStr ' -&gt; '];
end

for j = 1:length(prodMets)
if (j &gt; 1)
if (printFlag)
fprintf(fid,'+ ');
end
formulaStr = [formulaStr ' + '];
end
if (Sprod(j) &gt;= 1)
if (printFlag)
fprintf(fid,'%f %s ',Sprod(j),prodMets{j});
end
formulaStr = [formulaStr num2str(Sprod(j)) ' ' prodMets{j} ' '];
else
if (printFlag)
fprintf(fid,'%s ',prodMets{j});
end
formulaStr = [formulaStr prodMets{j} ' '];
end
else
if (printFlag)
fprintf(fid,'not in model');
end
formulaStr = 'NA';
end
if (printFlag)
if (isfield(model,'grRules'))
if (isempty(model.grRules{rxnID}))
fprintf('\tNA');
else
fprintf('\t%s',model.grRules{rxnID});
end
end
if (lineChangeFlag)
fprintf(fid,'\n');
end
end
formulas{i} = formulaStr;
end
formulas = formulas';
end

Appendix 2h.

MATLAB function findGaps for Mac OS.

function [fluxes gaps] = findGaps(model)

% This function was developed to find metabolic gaps in metabolic networks.
% model = a COBRA model
% for i = 1:length(model.c)
% model.c(i) = 1;
solution = optimizeCbModel(model);
model.c(i) = 0;
if i == 1
    fluxes = solution.f;
else
    fluxes = [fluxes; solution.f];
end
disp(i)
end

for i = 1:length(fluxes)
    if fluxes(i) == 0
        for j = 1:length(model.rxns)
            reactionName = model.rxns(j)
            metabolites = find(model.S(:,i) == 0)
            for k = 1:length(metabolites)
                model.S(metabolites(k),i) = 0
                solution = optimizeCbModel(model)
                if solution.f == 0
                    gaps = ['reactionName', model.metNames(metabolites(k)), 'Check'];
                    gaps = [gaps; reactionName, model.metNames(metabolites(k)), 'Check'];
                else
                    gaps = [gaps; reactionName, model.metNames(metabolites(k)), 'no_change'];
                end
            end
        end
        model.S = sbackup;
        model.c(i) = 0;
    end
end

Appendix 2i.

R function trackMetab for Mac OS.

track.metab <- function(compound, file = tk_choose.files(), column = "EQUATION"){
    ### Loading net
    model <- read.csv(file)
    ### Which column of the data frame contain the reactions?
    if (column == "EQUATION")
    } else {
        column <- select.list(names(model))
        column <- grep(column, names(model))
    }
    ### Which compound are you looking for?
    #compound <- "acetoAcetyl-CoA"
    ### Show all reactions involving this compound
    network <- model
    while (compound != ""){
        list.react <- paste(grep(compound, fixed = TRUE, network[,column], value = TRUE), grep(compound, fixed = TRUE, network[,column], value = FALSE))
        compound <- tk_select.list(list.react)
        #compound <- select.list(paste(grep(compound, fixed = TRUE, network[,column], value = TRUE), grep(compound, fixed = TRUE, network[,column], value = FALSE)))
        #compound <- gsub("=", "\+", compound)
        #compound <- gsub("=>", "\+", compound)
        divide <- select.list(c("Divide", "No, thanks"))
        if (divide == "Divide"){
            compound <- gsub("\+", "\+", compound)
        }
        ### Select the next compound you are looking for
        compound <- select.list(unlist(strsplit(compound, " + ", fixed = TRUE)))
    }
}
# Appendix 2j.

Combined genome-scale metabolic network of *Lactococcus lactis*.

<table>
<thead>
<tr>
<th>Gene code</th>
<th>Reactions</th>
<th>Ecs</th>
</tr>
</thead>
<tbody>
<tr>
<td>accA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin ⇔ ADP + phosphate + Malonyl-CoA</td>
<td>6.4.1.2</td>
</tr>
<tr>
<td>accB&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin ⇔ ADP + phosphate + Malonyl-CoA</td>
<td>6.4.1.2</td>
</tr>
<tr>
<td>accD&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin ⇔ ADP + phosphate + Malonyl-CoA</td>
<td>6.4.1.2</td>
</tr>
<tr>
<td>pycA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + Pyruvate + HCO(3) + biotin ⇔ ADP + phosphate + oxaloacetate</td>
<td>6.4.1.1</td>
</tr>
<tr>
<td>pycA&lt;sub&gt;2&lt;/sub&gt;</td>
<td>ATP + Pyruvate + HCO(3) + biotin ⇔ ADP + phosphate + oxaloacetate</td>
<td>6.4.1.1</td>
</tr>
<tr>
<td>pycA&lt;sub&gt;3&lt;/sub&gt;</td>
<td>ATP + Pyruvate + HCO(3) + biotin ⇔ ADP + phosphate + oxaloacetate</td>
<td>6.4.1.1</td>
</tr>
<tr>
<td>pycA&lt;sub&gt;4&lt;/sub&gt;</td>
<td>ATP + Pyruvate + HCO(3) + biotin ⇔ ADP + phosphate + oxaloacetate</td>
<td>6.4.1.1</td>
</tr>
<tr>
<td>carA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>2 ATP + L-glutamine + HCO(3) + H(2)O ⇒ 2 ADP + phosphate + L-glutamate + carbamoyl phosphate + 4 H(+)</td>
<td>6.3.5.5</td>
</tr>
<tr>
<td>carB&lt;sub&gt;1&lt;/sub&gt;</td>
<td>2 ATP + L-glutamine + HCO(3) + H(2)O ⇒ 2 ADP + phosphate + L-glutamate + carbamoyl phosphate + 4 H(+)</td>
<td>6.3.5.5</td>
</tr>
<tr>
<td>asnB&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + L-aspartate + L-glutamine + H(2)O ⇔ AMP + diphosphate + L-asparagine + L-glutamate</td>
<td>6.3.5.4</td>
</tr>
<tr>
<td>asnH&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + L-aspartate + L-glutamine ⇒ AMP + diphosphate + L-asparagine + L-glutamate</td>
<td>6.3.5.4</td>
</tr>
<tr>
<td>purL&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + 5'-phosphoribosylformylglycinamidine + L-glutamine + H(2)O ⇒ ADP + phosphate + 5'-phosphoribosylformylglycinamidine + L-glutamate</td>
<td>6.3.5.3</td>
</tr>
<tr>
<td>purQ&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + 5'-phosphoribosylformylglycinamidine + L-glutamine + H(2)O ⇒ ADP + phosphate + 5'-phosphoribosylformylglycinamidine + L-glutamate</td>
<td>6.3.5.3</td>
</tr>
<tr>
<td>guaA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + XMP + L-glutamine + H(2)O ⇒ AMP + diphosphate + GMP + L-glutamate</td>
<td>6.3.5.2</td>
</tr>
<tr>
<td>nadE&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + deamido-NAD(+) + L-glutamine + H(2)O ⇒ AMP + diphosphate + NAD(+) + L-glutamate</td>
<td>6.3.5.1</td>
</tr>
<tr>
<td>argG&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + L-citrulline + L-aspartate ⇒ AMP + diphosphate + L-argininocoeinate</td>
<td>6.3.4.5</td>
</tr>
<tr>
<td>purA&lt;sub&gt;1&lt;/sub&gt;</td>
<td>GTP + IMP + L-aspartate ⇔ GDP + phosphate + adenylosuccinate + 2 H(+)</td>
<td>6.3.4.4</td>
</tr>
<tr>
<td>fbs&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + formate + tetrahydrofolate ⇔ ADP + phosphate + 10-formyltetrahydrofolate</td>
<td>6.3.4.3</td>
</tr>
<tr>
<td>pyrG&lt;sub&gt;2&lt;/sub&gt;</td>
<td>ATP + UTP + L-glutamine + H(2)O ⇒ ADP + phosphate + CTP + L-glutamate</td>
<td>6.3.4.2</td>
</tr>
<tr>
<td>purD&lt;sub&gt;1&lt;/sub&gt;</td>
<td>ATP + 5-phospho-beta-D-riboylamine + glycine ⇔ ADP + phosphate + 5'-phosphoribosylglycinamidine</td>
<td>6.3.4.13</td>
</tr>
</tbody>
</table>
purM_1 ATP + 5′-phosphoribosylformylglycinamidine ⇒ ADP + phosphate + 1-(5′-phosphoribosyl)-5-aminomidazole 6.3.3.1
murD_1 ATP + UDP-N-acetylmuramoyl-L-alanine + D-glutamate ⇒ ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamate 6.3.2.9
murC_1 ATP + UDP-N-acetylmuramate + L-alanine ⇒ ADP + phosphate + UDP-N-acetylmuramoyl-L-alanine 6.3.2.8
purC_1 ATP + 1-(5-phosphoribosyl)-5-amino-4-imidazolecarboxylate + L-aspartate ⇒ ADP + phosphate + 1-(5-phosphoribosyl)-4-(N-succinocarboxamide)-5-aminimidazole 6.3.2.6
del_1 ATP + 2-D-alanine ⇒ ADP + phosphate + D-alanyl-D-alanine 6.3.2.4
folC_1 ATP + dihydropteroyl + L-glutamate ⇒ ADP + phosphate + dihydrofolate 6.3.2.17
murE_1 ATP + UDP-N-acetylmuramoyl-L-alanyl-D-glutamate + L-lysine ⇒ ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysine + H(2)O 6.3.2.13
murF_1 ATP + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysine + D-alanyl-D-alanine ⇒ ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine 6.3.2.10
glnA_1 ATP + L-glutamate + NH(3) ⇒ ADP + phosphate + L-glutamine 6.3.1.2
menE_1 ATP + 2-succinylbenzoate + CoA ⇒ AMP + diphosphate + 2-succinylbenzoyl-CoA 6.2.1.26
ywiC_1 chorismate ⇒ prephenate 5.4.99.5
ywiC_2 chorismate ⇒ prephenate 5.4.99.5
menF_1 chorismate ⇒ isochorismate 5.4.4.2
deoB_1 2-deoxy-D-ribose 1-phosphate ⇒ 2-deoxy-D-ribose 5-phosphate 5.4.2.7
deoB_2 D-ribose 5-phosphate ⇒ D-ribose 1-phosphate 5.4.2.7
pgmB_1 beta-D-glucose 1-phosphate ⇒ beta-D-glucose 6-phosphate 5.4.2.6
femD_1 D-glucose 1-phosphate ⇒ D-glucose 6-phosphate 5.4.2.2
femD_2 D-glucosamine 6-phosphate ⇒ D-glucosamine 1-phosphate 5.4.2.10
pmg_1 3-phospho-D-glycerate ⇒ 2-phospho-D-glycerate 5.4.2.1
yjbF_1 3-phospho-D-glycerate ⇒ 2-phospho-D-glycerate 5.4.2.1
yjI_1 3-phospho-D-glycerate ⇒ 2-phospho-D-glycerate 5.4.2.1
yebB_1 isopentenyl diphosphate ⇒ dimethylallyl diphosphate 5.3.3.2
pgiA_1 D-glucose 6-phosphate ⇒ D-fructose 6-phosphate 5.3.1.9
pgiA_2 D-glucose 6-phosphate ⇒ beta-D-glucose 6-phosphate 5.3.1.9
pni_1 D-mannose 6-phosphate ⇒ D-fructose 6-phosphate 5.3.1.8
rpiA_1 D-ribulose 5-phosphate ⇒ D-ribose 5-phosphate 5.3.1.6
xylA_1 D-xylene ⇒ D-xylulose 5.3.1.5
unk_10 D-galactose 6-phosphate ⇒ D-tagatose 6-phosphate 5.3.1.26
trpF_1 N-(5-phospho-beta-D-ribo)-anthranilate ⇒ 1-(2-carboxyphenylamino)-1-deoxy-D-ribulose-5-phosphate 5.3.1.24
hisA_1 N-(5′-phospho-D-ribosylformiminoo)-5-amino-1-(5′-phosphoribosyl)-4-imidazolecarboxamide ⇒ N-(5′-phospho-D-1′-ribosylformiminoo)-5-
amino-1-(5'-phosphoribosyl)-4-imidazolecarboxamide

uxaC_1  D-glucuronate <=> D-fructuronate  5.3.1.12
uxaC_2  D-galacturonate <=> D-tagaturonate  5.3.1.12
nagB_1  D-glucosamine 6-phosphate + H2O <=> D-fructose 6-phosphate + NH(3)  5.3.1.10
tpA_1  glycerone phosphate <=> D-glyceraldehyde 3-phosphate  5.3.1.1
galM_1  D-galactose <=> beta-D-galactose  5.1.3.3
xylM_1  D-glucose <=> beta-D-glucose  5.1.3.3
xylM_2  D-xylene <=> beta-D-xylose  5.1.3.3
xylM_3  L-arabinose <=> beta-L-arabinose  5.1.3.3
xylM_4  maltose <=> beta-maltose  5.1.3.3
galE_1  UDP-galactose <=> UDP-glucose  5.1.3.2
ycbD_1  UDP-galactose <=> UDP-glucose  5.1.3.2
cpsM_1  dTDP-4-dehydro-6-deoxy-D-glucose => dTDP-4-dehydro-6-deoxy-L-mannose  5.1.3.13
rpe_1  D-ribose 5-phosphate <=> D-xylulose 5-phosphate  5.1.3.1
murI_1  L-glutamate <=> D-glutamate  5.1.1.3
racD_1  L-aspartate <=> D-aspartate  5.1.1.13
dal_1  L-alanine <=> D-alanine  5.1.1.1
metC_1  cystathionine + H2O => L-homocysteine + NH(3) + Pyruvate  4.4.1.8
unk_29  cystathionine + H2O => L-cysteine + NH(3) + 2-oxobutanoate  4.4.1.1
purB_1  1-(5-phosphoribosyl)-4-(N-succinylcarboxamide)-5-aminomidazole-5-enzyme AICAR  4.3.3.2
purB_2  adenylosuccinate <=> fumarate + AMP  4.3.2.2
argH_1  L-argininosuccinate <=> fumarate + L-arginine  4.3.2.1
ilvA_1  L-threonine + H2O => 2-oxobutanoate + NH(3) + H2O  4.3.1.19
sdaA_1  L-serine <=> Pyruvate + NH(3)  4.3.1.17
sdaB_1  L-serine <=> Pyruvate + NH(3)  4.3.1.17
aroC_1  5-O-(1-carboxyvinyl)-3-phosphoshikimate => chorismate + phosphate  4.2.3.5
aroB_1  2-dehydro-3-deoxy-D-arabino-heptonate 7-phosphate => 3-dehydroquinone + phosphate  4.2.3.4
thrC_1  O-phospho-L-homoserine + H2O => L-threonine + phosphate  4.2.3.1
ilvD_1  2,3-dihydroxy-3-methylbutanoate => 3-methyl-2-oxobutanoate + H2O  4.2.1.9
ilvD_2  2,3-dihydroxy-3-methylpentanoate => 3-methyl-2-oxopentanoate + H2O  4.2.1.9
uxuA_1  D-maunonate <=> 2-dehydro-3-deoxy-D-gluconate + H2O  4.2.1.8
uxaA_1  D-alronate => 2-dehydro-3-deoxy-D-gluconate + H2O  4.2.1.7
aldB_1 2-acetolactate + H(+) <=> 2-acetoin + CO(2)

trpC_1 1-(2-carboxyphenylamino)-1-deoxy-D-ribulose-5-phosphate => 1-(indol-3-yl)glycerol 3-phosphate + CO(2) + H(2)O

pdc_1 phenylpyruvate => phenylacetaldehyde + CO(2)

yeaH_1 ATP + (R)-5-diphosphate => ADP + phosphate + isopentenyl diphosphate + CO(2)

unk_16 oxaloacetate => Pyruvate + CO(2)

pyrE_1 orotidine 5-phosphate => UMP + CO(2)

purE_1 1-(5-phosphoribosyl)-5-aminoimidazole + CO(2) <=> 1-(5-phosphoribosyl)-5-amino-4-imidazolecarboxylate

purK_1 1-(5-phosphoribosyl)-5-aminoimidazole + CO(2) <=> 1-(5-phosphoribosyl)-5-amino-4-imidazolecarboxylate

lysA_1 meso-2,6-diaminoheptanedioate => L-lysine + CO(2)

gadB_1 L-glutamate => 4-aminobutanoate + CO(2)

atpA 4 H(+)ext + ADP + phosphate <=> ATP + 3 H(+) + H(2)O

atpB 4 H(+)ext + ADP + phosphate <=> ATP + 3 H(+) + H(2)O

atpD 4 H(+)ext + ADP + phosphate <=> ATP + 3 H(+) + H(2)O

atpE 4 H(+)ext + ADP + phosphate <=> ATP + 3 H(+) + H(2)O

ATPp_1 4 H(+)ext + ADP + phosphate <=> ATP + 3 H(+) + H(2)O

hisI_1 1-(5-phospho-D-ribosyl)-AMP + H(2)O => 1-(5-phospho-D-ribosyl)-AMP + diphosphate

dut_1 dUTP + H(2)O => dUMP + diphosphate

ynbD_1 thiamine monophosphate + phosphate <=> thiamine diphosphate + H(2)O

DIPHOS diphosphate + H(2)O => 2 phosphate

tenA_1 thiamine + H(2)O => 4-amino-5-hydroxymethyl-2-methylpyrimidine + 4-methyl-5-(2-hydroxyethyl)-thiazole

cdd_1 cytidine + H(2)O => uridine + NH(3)

cdd_2 deoxycytidine + H(2)O => deoxyuridine + NH(3)

add_1 adenosine + H(2)O => inosine + NH(3)

add_2 deoxyadenosine + H(2)O => deoxyinosine + NH(3)

ribG_1 2,5-diamino-6-hydroxy-4-(5-phosphoribosylamino)pyrimidine + H(2)O => 5-amino-6-(5-phosphoribosylamino)uracil + NH(3)

ribA_1 GTP + 3 H(2)O => formate + 2,5-diamino-6-hydroxy-4-(5-phosphoribosylamino)pyrimidine + diphosphate

hisI_2 1-(5-phospho-D-ribosyl)-AMP + H(2)O => N-(5'-phospho-D-ribosylformimino)-5-amino-1-(5'-phosphoribosyl)-4-imidazolecarboxamide

folE_1 GTP + 2 H(2)O => formate + 2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropyridine triphosphate
dcdA_1  dCMP + H(2)O => dUMP + NH(3)  3.5.4.12  
purH_2  5-formamido-1-((5-phospho-D-ribo)nolyl)imidazole-4-carboxamide <=> IMP + H(2)O  3.5.4.10  
arcA_1  L-arginine + H(2)O => L-citrulline + NH(3)  3.5.3.6  
pyrC_1  N-carbamoyl-L-aspartate + H(+) <=> (S)-dihydroorotate + H(2)O  3.5.2.3  
unk_8  cysteamine + pantetheine <=> pantetheine + H(2)O  3.5.1.92  
nagA_1  N-acetyl-D-glucosamine 6-phosphate + H(2)O <=> D-glucosamine 6-phosphate + acetate  3.5.1.25  
pepV_1  N-succinyl-LL-2,6-diaminoheptanedioate + H(2)O => succinate + LL-2,6-diaminoheptanedioate  3.5.1.18  
argE_1  N2-acetyl-L-ornithine + H(2)O => acetate + L-ornithine  3.5.1.16  
anS_b_1  L-asparagine + H(2)O => L-aspartate + NH(3)  3.5.1.1  
bgIa_1  6-phospho-beta-D-glucose-(1,4)-D-glucose + H(2)O => D-glucose 6-phosphate + D-glucose  3.2.1.86  
bgIa_2  salicin 6-phosphate + H(2)O <=> beta-D-glucose 6-phosphate + 2-(hydroxymethyl)phenol  3.2.1.86  
bgIa_3  arbutin 6-phosphate + H(2)O <=> beta-D-glucose 6-phosphate + hydroquinone  3.2.1.86  
xynB_1  1,4-beta-D-xylan + H(2)O <=> 2 D-xylose  3.2.1.37  
xynB_2  xylobiose + H(2)O <=> 2 D-xylose  3.2.1.37  
unk_3  sucrose 6-phosphate + H(2)O <=> D-fructose + D-glucose 6-phosphate  3.2.1.26  
lacZ_1  lactose + H(2)O <=> D-galactose + D-glucose  3.2.1.23  
bgIh_2  p-Nitrophenyl-beta-D-glucopyranoside + H(2)O => p-nitrophenol + D-glucose  3.2.1.21  
agl_1  maltose + H(2)O => 2 D-glucose  3.2.1.20  
agl_2  sucrose + H(2)O => D-glucose + D-fructose  3.2.1.20  
dexA_1  alpha-dextrin => D-glucose  3.2.1.10  
amyY_3  glycogen + H(2)O => maltose  3.2.1.11  
amyY_4  glycogen + H(2)O => alpha-dextrin  3.2.1.11  
amyY_5  bglycogen + H(2)O => maltose  3.2.1.11  
amyY_6  bglycogen + H(2)O => alpha-dextrin  3.2.1.11  
amyl_3  glycogen + H(2)O => maltose  3.2.1.11  
amyl_4  glycogen + H(2)O => alpha-dextrin  3.2.1.11  
amyl_5  bglycogen + H(2)O => maltose  3.2.1.11  
amyl_6  bglycogen + H(2)O => alpha-dextrin  3.2.1.11  
acpD_1  ACP + H(2)O <=> pantetheine 4'-phosphate + apo-ACP  3.1.4.14  
acpD_1_6  ACP + H(2)O => pantetheine 4'-phosphate + apo-ACP  3.1.4.14  
ydiD_1  ACP + H(2)O => pantetheine 4'-phosphate + apo-ACP  3.1.4.14  
unk_65  adenosine 3',5'-bisphosphate + H(2)O => AMP + phosphate  3.1.3.7
nucA_1  AMP + H(2)O => adenosine + phosphate
nucA_10 dAMP + H(2)O => deoxyadenosine + phosphate
nucA_11 dGMP + H(2)O => deoxyguanosine + phosphate
nucA_2  IMP + H(2)O => inosine + phosphate
nucA_3  XMP + H(2)O => xanthosine + phosphate
nucA_4  GMP + H(2)O => guanosine + phosphate
nucA_5  CMP + H(2)O => cytidine + phosphate
nucA_6  UMP + H(2)O => uridine + phosphate
nucA_7  dTMP + H(2)O => thymidine + phosphate
nucA_8  dUMP + H(2)O => deoxyuridine + phosphate
nucA_9  dCMP + H(2)O => deoxycytidine + phosphate
serB_1   phosphoserine + H(2)O => L-serine + phosphate
pgpA_1   3-phosphatidylglycerol 1-phosphate + H(2)O => phosphatidylglycerol + phosphate
hisK_1   L-histidinol phosphate + H(2)O => L-histidinol + phosphate
flp_1    D-fructose 1,6-bisphosphate + H(2)O => D-fructose 6-phosphate + phosphate
acpS_1   CoA + apo-ACP <=> adenosine 3',5'-bisphosphate + ACP
acpS_1_du
psA_1    CDP-diacylglycerol + glycerol 3-phosphate <=> CMP + 3-phosphatidylglycerol 1-phosphate
psA_2    CDP-acylglycerol + glycerol 3-phosphate <=> CMP + lysothophatidylglycerol 1-phosphate
clsA_1   2 phosphatidylglycerol <=> cardiolipin + glycerol
clsB_1   2 phosphatidylglycerol <=> cardiolipin + glycerol
hasC_1   UTP + D-glucose 1-phosphate <=> diphosphate + UDP-glucose
cdsA_1   CTP + diacylglycerol 3-phosphate <=> diphosphate + GDP-diacylglycerol
cdsA_2   CTP + acylglycerol 3-phosphate <=> diphosphate + GDP-acylglycerol
tagD1_1   CTP + glycerol 3-phosphate + H(+) <=> diphosphate + GDP-glycerol
tagD2_1   CTP + glycerol 3-phosphate + H(+) <=> diphosphate + GDP-glycerol
kdtB_1   ATP + pantetheine 4'-phosphate <=> diphosphate + dephospho-CoA
glgC     D-glucose 1-phosphate + ATP <=> ADP-D-glucose + diphosphate
glgD     D-glucose 1-phosphate + ATP <=> ADP-D-glucose + diphosphate
rmL_1    dTTP + D-glucose 1-phosphate <=> diphosphate + dTDP-glucose
glmU_2   UTP + N-acetyl-alpha-D-glucosamine 1-phosphate <=> diphosphate + UDP-N-acetyl-D-glucosamine
riBC2
ATP + FMN => diphosphate + FAD

ylaE1
nicotinamide ribonucleotide + ATP => NAD(+) + diphosphate

ylaE2
nicotinate ribonucleotide + ATP <=> deamido-NAD(+) + diphosphate

gatT1
UTP + alpha-D-galactose 1-phosphate => diphosphate + UDP-galactose

folE2
dATP + D-ribose 5-phosphate => dAMP + 5-phospho-alpha-D-ribose 1-diphosphate + 2 H(+)

prsA1
ATP + D-ribose 5-phosphate => AMP + 5-phospho-alpha-D-ribose 1-diphosphate + 2 H(+)

prsB1
ATP + D-ribose 5-phosphate <=> AMP + 5-phospho-alpha-D-ribose 1-diphosphate + 2 H(+)

tmk1
ATP + dTMP <=> ADP + dTDP

tmk2
ATP + dUMP <=> ADP + dUDP

yeaB1
ATP + dTMP <=> ADP + dTDP

yeaB2
ATP + dUMP <=> ADP + dUDP

gmk1
ATP + GMP <=> ADP + GDP

gmk2
ATP + dGMP <=> ADP + dGDP

thiD1
ATP + 4-amino-2-methyl-5-phosphomethylpyrimidine => ADP + 4-amino-2-methyl-5-diphosphomethylpyrimidine

thiD2
ATP + 4-amino-2-methyl-5-phosphomethylpyrimidine => ADP + 4-amino-2-methyl-5-diphosphomethylpyrimidine

GDPk1
GDP + ATP <=> GTP + ADP

UDPk2
ATP + dUDP => ADP + dUTP

adk1
ATP + AMP <=> 2 ADP

adk2
ATP + dAMP <=> ADP + dADP

yebA1
ATP + (R)-5-phosphohexanaldehyde => ADP + (R)-5-diphosphohexanaldehyde

cmk1
ATP + CMP <=> ADP + CDP

cmk2
ATP + dCMP <=> ADP + dCDP

pyrH1
ATP + UMP <=> ADP + UDP

argB1
ATP + N-acetyl-L-glutamate => ADP + N-acetyl-L-glutamyl phosphate

pgk1
ADP + 3-phospho-D-glyceroyl phosphate <=> ATP + 3-phospho-D-glycerate

arcC1
ADP + carbamoyl phosphate <=> ATP + NH(3) + CO(2)

arcC2
ADP + carbamoyl phosphate <=> ATP + NH(3) + CO(2)

arcC3
ADP + carbamoyl phosphate <=> ATP + NH(3) + CO(2)

proB1
ATP + L-glutamate => ADP + L-gamma-glutamyl 5-phosphate

ackA1
ADP + Acetyl-phosphate <=> ATP + acetate

ackA2
ADP + Acetyl-phosphate <=> ATP + acetate

cb6p1
cellobiose 6-phosphate <=> 6-phospho-beta-D-glucoside-(1,4)-D-glucose

201
dukB_2  ATP + deoxyadenosine => ADP + dAMP

udk_4  ATP + deoxycytidine => ADP + dCMP

aroK_1  ATP + shikimate => ADP + 3-phosphoshikimate

celB_1  CELBext + phosphoenolpyruvate => cellobiose 6-phosphate + Pyruvate

fruA_1  FRUText + phosphoenolpyruvate => D-fructose 1-phosphate + Pyruvate

mflF_1  GLUCext + phosphoenolpyruvate => D-glucose 6-phosphate + Pyruvate

mflF_2  MNTLext + phosphoenolpyruvate => D-mannitol 1-phosphate + Pyruvate

ptbA_2  SALCext + phosphoenolpyruvate => salicin 6-phosphate + Pyruvate

ptbA_3  ARBText + phosphoenolpyruvate => arbutin 6-phosphate + Pyruvate

ptcA_1  CELBext + phosphoenolpyruvate => cellobiose 6-phosphate + Pyruvate

ptcB_1  CELBext + phosphoenolpyruvate => cellobiose 6-phosphate + Pyruvate

ptC_1  CELBext + phosphoenolpyruvate => cellobiose 6-phosphate + Pyruvate

ptnAB_1  GLUCext + phosphoenolpyruvate => D-glucose 6-phosphate + Pyruvate

ptnAB_2  MANNext + phosphoenolpyruvate => D-mannose 6-phosphate + Pyruvate

ptnAB_3  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

ptnAB_4  FRUText + phosphoenolpyruvate => D-fructose 1-phosphate + Pyruvate

ptnAB_5  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

ptnC_1  GLUCext + phosphoenolpyruvate => D-glucose 6-phosphate + Pyruvate

ptnC_2  MANNext + phosphoenolpyruvate => D-mannose 6-phosphate + Pyruvate

ptnC_3  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

ptnC_4  FRUText + phosphoenolpyruvate => D-fructose 1-phosphate + Pyruvate

ptnC_5  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

ptnD_1  GLUCext + phosphoenolpyruvate => D-glucose 6-phosphate + Pyruvate

ptnD_2  MANNext + phosphoenolpyruvate => D-mannose 6-phosphate + Pyruvate

ptnD_3  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

ptnD_4  FRUText + phosphoenolpyruvate => D-fructose 1-phosphate + Pyruvate

ptnD_5  GLUMext + phosphoenolpyruvate => D-glucosamine 6-phosphate + Pyruvate

yleD_1  SUCRext + phosphoenolpyruvate => sucrose 6-phosphate + Pyruvate

yleD_2  TRHLext + phosphoenolpyruvate => trehalose 6-phosphate + Pyruvate

bacA_1  ADP + undecaprenyl phosphate + 2 H(+) <=> ATP + undecaprenol

galK_1  ATP + D-galactose => ADP + alpha-D-galactose 1-phosphate
2.7.1.56

2.7.1.50

2.7.1.49

2.7.1.48

2.7.1.48

2.7.1.45

2.7.1.40

2.7.1.39

2.7.1.38

2.7.1.37

2.7.1.36

2.7.1.35

2.7.1.34

2.7.1.33

2.7.1.32

2.7.1.31

2.7.1.30

2.7.1.29

2.7.1.28

2.7.1.27

2.7.1.26

2.7.1.25

2.7.1.24

2.7.1.23

2.7.1.22

2.7.1.21

2.7.1.20

2.7.1.19

2.7.1.18

2.7.1.17

2.7.1.16

2.7.1.15

2.7.1.14

2.7.1.13

2.7.1.12

2.7.1.11

2.7.1.10

2.7.1.9

2.7.1.8

2.7.1.7

2.7.1.6

2.7.1.5

2.7.1.4

2.7.1.3

2.7.1.2

2.7.1.1

2.6.1.60

2.6.1.57

2.6.1.56

2.6.1.55

2.6.1.54

2.6.1.53

2.6.1.52

2.6.1.51

2.6.1.50

2.6.1.49
araT_7  phenylpyruvate + L-glutamate ⇄ L-phenylalanine + 2-oxoglutarate  2.6.1.57
serC_1  phosphoserine + 2-oxoglutarate ⇄ 3-phosphohydroxyphenylpyruvate + L-glutamate  2.6.1.52
unk_22  L-methionine + 2-oxoglutarate ⇄ 4-methylthio-2-oxobutyrate + L-glutamate  2.6.1.5
bcaT_1  L-valine + 2-oxoglutarate ⇄ 3-methyl-2-oxobutanoate + L-glutamate  2.6.1.42
bcaT_2  L-isoleucine + 2-oxoglutarate ⇄ 3-methyl-2-oxopentanoate + L-glutamate  2.6.1.42
bcaT_3  L-leucine + 2-oxoglutarate ⇄ 4-methyl-2-oxopentanoate + L-glutamate  2.6.1.42
araT_10  L-phenylalanine + indol-3-pyruvate ⇄ phenylpyruvate + L-tryptophan  2.6.1.28
araT_16  L-tryptophan + phenylpyruvate ⇄ indol-3-pyruvate + L-phenylalanine  2.6.1.28
araT_13  L-tryptophan + 2-oxoglutarate ⇄ indol-3-pyruvate + L-glutamate  2.6.1.27
unk_18  Pyruvate + L-glutamate ⇄ L-alanine + 2-oxoglutarate  2.6.1.2
unk_31  N-succinyl-L-2-amino-6-oxoheptanedioate + L-glutamate ⇄ N-succinyl-LL-2,6-diaminoheptanedioate + 2-oxoglutarate  2.6.1.17
glmS_1  L-glutamine + D-fructose 6-phosphate ⇄ L-glutamate + D-glucosamine 6-phosphate  2.6.1.16
argD_1  N-acetyl-L-glutamate 5-semialdehyde + L-glutamate ⇄ N2-acetyl-L-ornithine + 2-oxoglutarate  2.6.1.11
aspB_1  oxaloacetate + L-glutamate ⇄ L-aspartate + 2-oxoglutarate  2.6.1.1
ribH_1  2,6,7-dimethyl-8-(1-D-ribofuranosyl)imidazole ⇄ riboflavin + 4-(1-D-ribofuranosyl)-5-amino-2,6-dihydroxypyrimidine  2.5.1.9
murA1_1  phosphoenolpyruvate + UDP-N-acetyl-D-glucosamine ⇄ phosphate + UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine  2.5.1.7
murA2_1  phosphoenolpyruvate + UDP-N-acetyl-D-glucosamine ⇄ phosphate + UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine  2.5.1.7
metK_1  ATP + L-methionine + H2O ⇄ phosphate + diphasphate + S-adenosyl-L-methionine + H(+)  2.1.6
cysD_1  O-acetyl-L-homoserine + methanethiol ⇄ L-methionine + acetate  2.5.1.49
cysD_2  O-acetyl-L-homoserine + H2S ⇄ L-homocysteine + acetate  2.5.1.49
metB1_1  O-succinyl-L-homoserine + L-cysteine ⇄ cystathionine + succinate  2.5.1.48
metB2_1  O-succinyl-L-homoserine + L-cysteine ⇄ cystathionine + succinate  2.5.1.48
cysK_1  O-acetyl-L-serine + H2S ⇄ L-cysteine + acetate  2.5.1.47
cysM_1  O-acetyl-L-serine + H2S ⇄ L-cysteine + acetate  2.5.1.47
uppS_1  8 isopentenyl diphosphate + trans-2-farnesyl diphosphate ⇄ undecaprenyl diphosphate + 8 diphosphate  2.5.1.31
ispB_1  4 isopentenyl diphosphate + trans-2-farnesyl diphosphate ⇄ trans-2-heptaprenyl diphosphate + 4 diphosphate  2.5.1.30
thiE_1  4-amino-2-methyl-5-diphosphomethylpyrimidine + 4-methyl-5-(2-phosphoethenyl)-thiazole ⇄ diphosphate + thiamine monophosphate  2.5.1.3
aroA_1  phosphoenolpyruvate + 3-phosphoshikimate ⇄ phosphate + 5-O-(1-carboxyvinyl)-3-phosphoshikimate  2.5.1.19
folP_1  2-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine diphosphate + 4-amino-benzamide ⇄ diphosphate + dihydropteroylglutamate  2.5.1.15
dimethylallyl diphosphate + isopentenyl diphosphate => geranyl diphosphate + diphosphate 2.5.1.10

geranyl diphosphate + isopentenyl diphosphate => diphosphate + trans-2,4-dihydroxy-3-hydroxy-2,4-dimethyl-3-butenyl diphosphate 2.5.1.10

3,4-dihydroxy-2-butanone 4-phosphate + 4-(1-D-ribitylamino)-5-amino-2,6-dihydroxypyrimidine => 6,7-dimethyl-8-(1-D-ribofuranosyl)benzimidazol-2(3H)-one + phosphate 2.5.1.10

ADPK_2

dADP + ATP <= dATP + ADP 2.4.7.6

uracil + 5-phospho-alpha-D-ribose 1-diphosphate <= UMP + diphosphate 2.4.2.9

IMP + diphosphate <= hypoxanthine + 5-phospho-alpha-D-ribose 1-diphosphate 2.4.2.8

GMP + diphosphate <= guanine + 5-phospho-alpha-D-ribose 1-diphosphate 2.4.2.8

IMP + diphosphate <= hypoxanthine + 5-phospho-alpha-D-ribose 1-diphosphate 2.4.2.8

GMP + diphosphate <= guanine + 5-phospho-alpha-D-ribose 1-diphosphate 2.4.2.8

adenine + 5-phospho-alpha-D-ribose 1-diphosphate <= AMP + diphosphate 2.4.2.7

uridine + phosphate <= uracil + D-ribose 1-phosphate 2.4.2.7

deoxyuridine + phosphate <= uracil + 2-deoxy-D-ribose 1-phosphate 2.4.2.3

xanthine + 5-phospho-alpha-D-ribose 1-diphosphate <= XMP + diphosphate 2.4.2.22

cytidine + phosphate <= cytosine + D-ribose 1-phosphate 2.4.2.2

deoxycytidine + phosphate <= cytosine + D-ribose 1-phosphate 2.4.2.2

thymine + 2-deoxy-D-ribose 1-phosphate <= thymidine + phosphate 2.4.2.2

anthranilate + 5-phospho-alpha-D-ribose 1-diphosphate <= N-(5-phospho-beta-D-ribofuranosyl)-anthranilate + diphosphate 2.4.2.18

ATP + 5-phospho-alpha-D-ribose 1-diphosphate <= 1-(5-phospho-D-ribosyl)-ATP + diphosphate 2.4.2.17

L-glutamine + 5-phospho-alpha-D-ribose 1-diphosphate + H(2)O => 5-phospho-beta-D-ribofuranosylamine + diphosphate + L-glutamate 2.4.2.14

nicotinate + 5-phospho-alpha-D-ribose 1-diphosphate + H(+) => nicotinate ribonucleotide + diphosphate 2.4.2.11

orotate + 5-phospho-alpha-D-ribose 1-diphosphate <= orotidine 5-phosphate + diphosphate 2.4.2.10

guanosine + phosphate <= guanine + D-ribose 1-phosphate 2.4.2.1

inosine + phosphate <= hypoxanthine + D-ribose 1-phosphate 2.4.2.1

deoxyguanosine + phosphate <= guanine + D-ribose 1-phosphate 2.4.2.1

deoxyinosine + phosphate <= hypoxanthine + D-ribose 1-phosphate 2.4.2.1

xanthosine + phosphate <= xanthine + D-ribose 1-phosphate 2.4.2.1

N-(5-phospho-D-ribofuranosyl)-5-amino-1-(5-phosphoribosyl)-4-imidazolcarboxamid e+ L-glutamine <= AICAR + D-erythro-1-(imidazol-4-y1)glycerol 3-phosphate + L-glutamate 2.4.2.1

hisF_1
N-(5'-phospho-D-1'-ribulosylformiminono)-5-amino-1-(5'-phosphoribosyl)-4-imidazolcarboxamide + L-glutamine ⇔ AICAR + D-erythro-1-(imidazol-4-yl)glycerol 3-phosphate + L-glutamate

mapA_1 maltose + phosphate ⇔ D-glucose + beta-D-glucose 1-phosphate


glgA ADP-D-glucose ⇔ ADP + glycogen + H(+)
glgB glycogen ⇔ bglycogen
glgP_1 glycogen + phosphate ⇔ D-glucose 1-phosphate
glgP_2 bglycogen + phosphate ⇔ D-glucose 1-phosphate

Acetyl-CoA + 3-methyl-2-oxobutanoate + H(2)O ⇔ 2-isopropylmalate + CoA

Acetyl-CoA + H(2)O + Acetoacetyl-CoA ⇔ (S)-3-hydroxy-3-methylglutaryl-CoA + CoA + H(+)

Acetyl-CoA + H(2)O + oxaloacetate ⇔ citrate + CoA

2 Acetyl-CoA ⇔ CoA + Acetoacetyl-CoA

(R)-3-hydroxyhexanoyl-(acp) ⇔ trans-hex-2-enoyl-(acp) + H(2)O

(R)-3-hydroxyoctanoyl-(acp) ⇔ trans-oct-2-enoyl-(acp) + H(2)O

(R)-3-hydroxydecanoyl-(acp) ⇔ trans-dec-2-enoyl-(acp) + H(2)O

(R)-3-hydroxytetradecanoyl-(acp) ⇔ trans-tetradec-2-enoyl-(acp) + H(2)O

(R)-3-hydroxypalmitoyl-(acp) ⇔ trans-hexadec-2-enoyl-(acp) + H(2)O

Acetyl-CoA + phosphate ⇔ CoA + Acetyl-phosphate

Acetyl-CoA + maltose ⇔ CoA + acetyl-maltose

CoA + Pyruvate ⇔ Acetyl-CoA + formate

succinyl-CoA + L-homoserine ⇔ CoA + O-succinyl-L-homoserine

Acetyl-(acp) + Malonyl-(acp) ⇔ Acetoacetyl-(acp) + CO(2) + ACP

Butyryl-(acp) + Malonyl-(acp) ⇔ 3-Oxohexanoyl-(acp) + CO(2) + ACP

Hexanoyl-(acp) + Malonyl-(acp) ⇔ 3-Oxoctanoyl-(acp) + CO(2) + ACP

Octanoyl-(acp) + Malonyl-(acp) ⇔ 3-octodecanoyl-(acp) + CO(2) + ACP

Decanoyl-(acp) + Malonyl-(acp) ⇔ 3-Oxodecanoyl-(acp) + CO(2) + ACP

Dodecanoyl-(acp) + Malonyl-(acp) ⇔ 3-Oxotetradecanoyl-(acp) + CO(2) + ACP

Tetradecanoyl-(acp) + Malonyl-(acp) ⇔ 3-oxohexadecanoyl-(acp) + CO(2) + ACP

Malonyl-CoA + ACP ⇔ Malonyl-(acp) + CoA
207

fabH_1  Acetyl-CoA + ACP ⇌ Acetyl-(acp) + CoA  2.3.1.38
argJ_2  N2-acetyl-L-ornithine + L-glutamate ⇒ L-ornithine + N-acetyl-L-glutamate  2.3.1.35
metX_1  L-homoserine + Acetyl-CoA ⇒ O-acetyl-L-homoserine + CoA  2.3.1.31
cysE_1  Acetyl-CoA + L-serine ⇌ CoA + O-acetyl-L-serine  2.3.1.30
glmU_1  D-glucosamine 1-phosphate + Acetyl-CoA ⇒ N-acetyl-alpha-D-glucosamine 1-phosphate + CoA  2.3.1.157
pydC_1  CoA + S-acetyl-dihydrolipoamide ⇌ Acetyl-CoA + dihydrolipoamide  2.3.1.12
ychH_1  succinyl-CoA + 2,3,4,5-tetrahydropyridine-2,6-dicarboxylate + H(2)O ⇒ CoA + N-succinyl-L-2-amino-6-oxoheptanoate  2.3.1.117
argJ_1  Acetyl-CoA + L-glutamate ⇒ CoA + N-acetyl-L-glutamate  2.3.1.1
menD_1  isochorismate + 2-oxoglutarate ⇒ 2-succinyl-6-hydroxy-2,4-cyclohexadiene-1-carboxylate + Pyruvate + CO2  2.2.1.9
als_1  2 Pyruvate + H(+) ⇒ 2-acetolactate + CO2  2.2.1.6
ilvB_1  2 Pyruvate + H(+) ⇒ 2-acetolactate + CO2  2.2.1.6
ilvB_2  Pyruvate + 2-oxobutanoate + H(+) ⇒ 2-aceto-2-hydroxybutanoate + CO2  2.2.1.6
ilvN_1  2 Pyruvate + H(+) ⇒ 2-acetolactate + CO2  2.2.1.6
unk_2  sedoheptulose 7-phosphate + D-glyceraldehyde 3-phosphate ⇌ D-fructose 6-phosphate + D-erythrose 4-phosphate  2.2.1.2
tkt_1  D-ribose 5-phosphate + D-xylulose 5-phosphate ⇌ sedoheptulose 7-phosphate + D-glyceraldehyde 3-phosphate  2.2.1.1
tkt_2  D-erythrose 4-phosphate + D-xylulose 5-phosphate + thiamine diphosphate ⇌ D-fructose 6-phosphate + D-glyceraldehyde 3-phosphate  2.2.1.1
argF_1  carbamoyl phosphate + L-ornithine ⇌ phosphate + L-citrulline  2.1.3.3
pyrB_1  carbamoyl phosphate + L-aspartate ⇒ phosphate + N-carbamoyl-L-aspartate + H(+)  2.1.3.2
purH_1  10-formyltetrahydrofolate + AICAR ⇒ tetrahydrofolate + 5-formamidomethyl-D-ribosylimidazole-4-carboxamide  2.1.2.3
purN_1  10-formyltetrahydrofolate + 5'-phosphoribosylglycinamide ⇒ tetrahydrofolate + 5'-phosphoribosylformylglycinamide  2.1.2.2
panB_1  3-methyl-2-oxobutanoate + 5,10-methylenetetrahydrofolate + H2O ⇒ tetrahydrofolate + 2-dehydropantoate  2.1.2.11
glyA_1  5,10-methylenetetrahydrofolate + glycine + H2O ⇌ tetrahydrofolate + L-serine  2.1.2.1
thyA_1  5,10-methylenetetrahydrofolate + dUMP ⇌ dihydrofolate + dTMP  2.1.1.45
metE_1  5-methyltetrahydrofolate + L-homocysteine ⇌ tetrahydrofolate + L-methionine  2.1.1.13
ubiE_1  2-demethylmenaquinone + S-adenosyl-L-methionine ⇒ S-adenosyl-L-homocysteine + menaquinone + H(+)  2.1.1.1
taxB1_1  NADPH + oxidized thioredoxin ⇒ NADP(+) + reduced thioredoxin  1.8.1.9
taxB2_1  NADPH + H(+) + oxidized thioredoxin ⇒ NADP(+) + reduced thioredoxin  1.8.1.9
gshR_1  NADPH + H(+) + oxidized glutathione ⇒ NADP(+) + 2 glutathione  1.8.1.7
pdhD_1  dihydrolipoamide + NAD(+) ⇌ lipoamide + NADH + H(+)

noxA_1  NADH + H(+) + O(2) ⇌ H(2)O(2) + NAD(+)

noxB_1  NADH + H(+) + O(2) ⇌ H(2)O(2) + NAD(+)

guaC_1  NADPH + GMP ⇌ NADP(+) + IMP + NH(3) + H(+)

unk_4  menaquinol ⇌ menaquinone + 2 H(+)

folD_1  5,10-methylenetetrahydrofolate + H(2)O ⇌ 5,10-methenyltetrahydrofolate + H(+)

noxA_1  NADH + H(+) + O(2) ⇌ H(2)O(2) + NAD(+)

noxB_1  NADH + H(+) + O(2) ⇌ H(2)O(2) + NAD(+)

guaC_1  NADPH + GMP ⇌ NADP(+) + IMP + NH(3) + H(+)

unk_4  menaquinol ⇌ menaquinone + 2 H(+)

folD_1  5,10-methylenetetrahydrofolate + H(2)O ⇌ 5,10-methenyltetrahydrofolate + NADP(+)

folD_2  5,10-methylenetetrahydrofolate + FADH(2) ⇌ 5-methyltetrahydrofolate + FAD

proC_1  1-pyrroline-5-carboxylate + NADPH + H(+) ⇌ L-proline + NADP(+)

gltB_1  L-glutamine + 2-oxoglutarate + NADPH + H(+) ⇌ 2 L-glutamate + NADP (+)

gltD_1  L-glutamine + 2-oxoglutarate + NADPH + H(+) ⇌ 2 L-glutamate + NADP (+)

pydA_1  (S)-dihydroorotate + O(2) ⇌ orotate + H(2)O(2)

pydB_1  (S)-dihydroorotate + O(2) ⇌ orotate + H(2)O(2)

pydB_2  (S)-dihydroorotate + NAD(+) ⇌ orotate + NADH + H(+)

FabI_1  But-2-enoyl-(acp) + NADH + H(+) ⇌ Butyryl-(acp) + NAD(+)

FabI_2  trans-2-enoyl-(acp) + NADH + H(+) ⇌ Hexanoyl-(acp) + NAD(+)

FabI_3  trans-2-enoyl-(acp) + NADH + H(+) ⇌ Octanoyl-(acp) + NAD(+)

FabI_4  trans-2-enoyl-(acp) + NADH + H(+) ⇌ Decanoyl-(acp) + NAD(+)

FabI_5  trans-2-enoyl-(acp) + NADH + H(+) ⇌ Dodecanoyl-(acp) + NAD(+)

FabI_6  trans-tetradec-2-enoyl-(acp) + NADH + H(+) ⇌ Tetradecanoyl-(acp) + NAD(+)

FabI_7  trans-hexadec-2-enoyl-(acp) + NADH + H(+) ⇌ Hexadecanoyl-(acp) + NAD(+)

dapB_1  2,3-dihydroliposinate + NADPH + H(+) ⇌ 2,3,4,5-tetrahydropyridine-2,6-dicarboxylate + NADP(+)

tyrA_1  prephenate + NAD(+) ⇌ 4-hydroxyphenylpyruvate + CO(2) + NADH + H(+)

unk_4  2-oxoglutarate + NAD(+) + CoA ⇌ succinyl-CoA + NADH + H(+) + CO(2)

pdhA_1  Pyruvate + lipoamide + H(+) ⇌ S-acetylhydrodipicolinate + CO(2)

pdhB_1  Pyruvate + lipoamide + H(+) ⇌ S-acetylhydrodipicolinate + CO(2)

poxL_1  Pyruvate + phosphate + O(2) + H(2)O ⇌ Acetyl-phosphate + CO(2) + H(2)O(2)

2.3.1.26

1.3.1.12

1.2.4.2

1.2.4.1

1.2.4.1

1.2.3.3
\[
\begin{align*}
\text{proA}_1 & : \text{L-gamma-glutamyl 5-phosphate + NADPH + H}^+ \Leftrightarrow \text{L-glutamate 5-semialdehyde + phosphate + NADP}^+ \\
\text{argC}_1 & : \text{N-acetyl-5-glutamyl phosphate + NADPH + H}^+ \Leftrightarrow \text{N-acetyl-L-glutamate 5-semialdehyde + NADP}^+ + \text{phosphate} \\
\text{gapA}_1 & : \text{D-glyceraldehyde 3-phosphate + phosphate + NAD}^+ \Leftrightarrow \text{3-phospho-D-glyceroyl phosphate + NADH + H}^+ \\
\text{gapB}_1 & : \text{D-glyceraldehyde 3-phosphate + phosphate + NAD}^+ \Leftrightarrow \text{3-phospho-D-glyceroyl phosphate + NADH + H}^+ \\
\text{asd}_1 & : 4\text{-phospho-L-aspartate + NADPH} \Leftrightarrow \text{L-aspartate 4-semialdehyde + phosphate + NADP}^+ \\
\text{adhE}_1 & : \text{Acetyl-CoA + NADH + H}^+ \Leftrightarrow \text{acetaldehyde + CoA + NAD}^+ \\
\text{nrdD}_1 & : \text{ATP + reduced thioredoxin} \Rightarrow \text{dATP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdD}_2 & : \text{GTP + reduced thioredoxin} \Rightarrow \text{dGTP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdD}_3 & : \text{CTP + reduced thioredoxin} \Rightarrow \text{dCTP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdD}_4 & : \text{UTP + reduced thioredoxin} \Rightarrow \text{dUTP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdE}_1 & : \text{ADP + reduced thioredoxin} \Rightarrow \text{dADP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdE}_2 & : \text{GDP + reduced thioredoxin} \Rightarrow \text{dGDP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdE}_3 & : \text{CDP + reduced thioredoxin} \Rightarrow \text{dCDP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{nrdE}_4 & : \text{UDP + reduced thioredoxin} \Rightarrow \text{dUDP + oxidized thioredoxin} + \text{H}_2\text{O} \\
\text{sodA}_1 & : \text{2 superoxide + 2 H}^+ \Leftrightarrow \text{O}_2 + \text{H}_2\text{O}_2 \\
\text{gpo}_1 & : \text{2 glutathione + H}_2\text{O}_2 \Rightarrow \text{oxidized glutathione + 2 H}_2\text{O} \\
\text{serA}_1 & : \text{3-phospho-D-glycerate + NAD}^+ \Leftrightarrow \text{3-phosphohydroxypyruvate + NADH + H}^+ \\
\text{gpdA}_1 & : \text{glycerone phosphate + NADPH + H}^+ \Leftrightarrow \text{glycerol 3-phosphate + NADP}^+ \\
\text{mvaA}_1 & : \text{(S)-3-hydroxy-3-methylglutaryl-CoA + 2 NADPH + 2 H}^+ \Rightarrow \text{(R)-mevalonate + CoA + 2 NADP}^+ \\
\text{ilvC}_1 & : \text{2-acetolactate + NADPH + H}^+ \Leftrightarrow \text{2,3-dihydroxy-3-methylbutanoate + NADP}^+ \\
\text{ilvC}_2 & : \text{2-aceto-2-hydroxybutanoate + NADPH + H}^+ \Leftrightarrow \text{2,3-dihydroxy-3-methylpentanoate + NADP}^+ \\
\text{leuB}_1 & : \text{3-isopropylmalate + NAD}^+ \Rightarrow \text{4-methyl-2-oxopentanoate + NADH + H}^+ + \text{CO}_2 \\
\end{align*}
\]
uxAB\_1  D-tagaturonate + NADH + H(+) ⇌ D-altronate + NAD(+)  1.1.1.58
uxuB\_1  D-fructuronate + NADH + H(+) ⇌ D-mannonate + NAD(+)  1.1.1.57
uxuB\_2  D-fructuronate + NADH + H(+) ⇌ D-mannonate + NAD(+)  1.1.1.57
butA\_1  diacetyl + NADH + H(+) ⇌ 2-acetoin + NAD(+)  1.1.1.5
zwf\_1  6-phospho-D-gluconate + NADP(+) ⇌ D-ribulose-5-phosphate + CO(2) + NADPH + H(+)  1.1.1.44
GND\_1  6-phospho-D-gluconate + NADP(+) ⇌ D-ribulose-5-phosphate + CO(2) + NADPH + H(+)  1.1.1.44
gntZ\_1  (S)-malate + NAD(+) ⇌ Pyruvate + CO(2) + NADH + H(+)  1.1.1.38
mLE\_1  (S)-malate → (S)-lactate + CO(2)  1.1.1.38
maeP\_1  MALA\text{ext} + H(+)\text{ext} ⇌ (S)-malate + H(+)  1.1.1.38
hom\_1  L-aspartate 4-semialdehyde + NADPH + H(+) ⇌ L-homoserine + NADP(+)  1.1.1.3
tha\_1  ATP + L-aspartate → ADP + 4-phospho-L-aspartate  1.1.1.3
ldh\_1  Pyruvate + NADH + H(+) ⇌ (S)-lactate + NAD(+)  1.1.1.27
ldhB\_1  Pyruvate + NADH + H(+) ⇌ (S)-lactate + NAD(+)  1.1.1.27
ldhX\_1  Pyruvate + NADH + H(+) ⇌ (S)-lactate + NAD(+)  1.1.1.27
aroE\_1  3-dehydroshikimate + NADPH + H(+) ⇌ shikimate + NADP(+)  1.1.1.25
unk\_35  phenylpyruvate + NADH + H(+) → phenyllactate + NAD(+)  1.1.1.237
hisD\_1  L-histidinol + 2 NAD(+) + H(2)O → L-histidine + 2 NADH + 2 H(+)  1.1.1.23
unh\_33  4-hydroxyphenylpyruvate + NADH ⇌ 4-hydroxyphenyllactate + NAD(+)  1.1.1.222
gua\_1  IMP + NAD(+) + H(2)O → XMP + NADH + H(+)  1.1.1.205
ribG\_2  5-amino-6-(5-phosphoribosylamino)uracil + NADPH + H(+) ⇌ 5-amino-6-(5-phosphoribitylamino)uracil + NADP(+)  1.1.1.193
rflD\_1  D-mannitol 1-phosphate + NAD(+) ⇌ D-fructose 6-phosphate + NADH + H(+)  1.1.1.17
panE\_1  (R)-pantoate + NADP(+) ⇌ 2-dehydropantoate + NADPH + H(+)  1.1.1.169
murB\_1  UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine + NADPH + H(+)  
\text{⇒}  UDP-N-acetylmuramate + NADP(+)  1.1.1.158
rmC\_1  dTDP-4-dehydro-6-deoxy-L-mannose + NADPH + H(+) ⇌ dTDP-rhamnose + NADP(+)  1.1.1.133
FabG\_1  Acetoacetyl-(acp) + NADPH + H(+) ⇌ (R)-3-hydroxybutanoyl-(acp) + NADP(+)  1.1.1.100
FabG\_2  3-Oxohexanoyl-(acp) + NADPH + H(+) ⇌ (R)-3-hydroxyhexanoyl-(acp) + NADP(+)  1.1.1.100
FabG\_3  3-Oxocaproyl-(acp) + NADPH + H(+) ⇌ (R)-3-hydroxyoctanoyl-(acp) + NADP(+)  1.1.1.100
**FabG_4**

3-oxodecanoyl-(acp) + NADPH ⇄ (R)-3-hydroxydecanoyl-(acp) + NADP(+)

1.1.1.100

**FabG_5**

3-Oxodecanoyl-(acp) + NADPH + H(+) ⇄ (R)-3-hydroxydecanoyl-(acp) + NADP(+)

1.1.1.100

**FabG_6**

3-Oxotetradecanoyl-(acp) + NADPH + H(+) ⇄ (R)-3-hydroxytetradecanoyl-(acp) + NADP(+)

1.1.1.100

**FabG_7**

3-oxohexadecanoyl-(acp) + NADPH + H(+) ⇄ (R)-3-hydroxyhexadecanoyl-(acp) + NADP(+)

1.1.1.100

**adhA_1**

acetaldehyde + NADH + H(+) ⇄ ethanol + NAD(+)

1.1.1.1

**thrA_2**

ATP + L-aspartate ⇄ ADP + 4-phospho-L-aspartate

2.7.2.4

**araT_12**

L-phenylalanine + 4-hydroxyphenylpyruvate ⇄ phenylpyruvate + L-tyrosine

**araT_14**

L-tryptophan + 4-methyl-2-oxopentanoate ⇄ indol-3-pyruvate + L-leucine

**araT_15**

L-tryptophan + 3-methyl-2-oxopentanoate ⇄ indol-3-pyruvate + L-isoleucine

**araT_17**

L-tryptophan + 4-methylthio 2-oxobutyrate ⇄ 4-hydroxyphenylpyruvate + L-methionine

**araT_18**

L-tryptophan + 4-hydroxyphenylpyruvate ⇄ indol-3-pyruvate + L-tyrosine

**araT_2**

L-tyrosine + 4-methyl-2-oxopentanoate ⇄ 4-hydroxyphenylpyruvate + L-leucine

**araT_3**

L-tyrosine + 3-methyl-2-oxopentanoate ⇄ 4-hydroxyphenylpyruvate + L-isoleucine

**araT_4**

L-tyrosine + phenylpyruvate ⇄ 4-hydroxyphenylpyruvate + L-phenylalanine

**araT_5**

L-tyrosine + 4-methylthio 2-oxobutyrate ⇄ 4-hydroxyphenylpyruvate + L-methionine

**araT_6**

L-tyrosine + indol-3-pyruvate ⇄ 4-hydroxyphenylpyruvate + L-tryptophan

**araT_8**

L-phenylalanine + 4-methyl-2-oxopentanoate ⇄ phenylpyruvate + L-leucine

**araT_9**

L-phenylalanine + 3-methyl-2-oxopentanoate ⇄ phenylpyruvate + L-isoleucine

**aspC_1**

oxaloacetate + L-glutamate ⇄ L-aspartate + 2-oxoglutarate

**bPOLYS**

5.5 UDP-glucose + UDP-galactose + 5.6 dTDP-rhamnose ⇄ POLYS + 6.5 UDP + 5.6 dTDP

**c140d_1**

myristic ⇄ diphtosate

**c140s_1**

Acetyl-(acp) + 6 Malonyl-(acp) + 12 NADPH + 12 H(+) ⇄ myristoyl-(acp) + 12 NADP(+). + 6 CO(2) + 6 ACP + 6 H(2)O

**c141s_1**

Acetyl-(acp) + 6 Malonyl-(acp) + 11 NADPH + 11 H(+) ⇄ tetradecenoyl-(acp) + 11 NADP(+). + 6 CO(2) + 6 ACP + 6 H(2)O

**c160d_1**

palmitate + ATP + 8 CoA + 8 FAD + 8 NAD(+) ⇄ AMP + diphosphate + 8 FADH(2) + 8 NADH + 8 H(+) + 8 Acetyl-CoA

**c160s_1**

Acetyl-(acp) + 7 Malonyl-(acp) + 14 NADPH + 14 H(+) ⇄ Hexadecanoyl-(acp) + 14 NADP(+). + 7 CO(2) + 7 ACP + 7 H(2)O

**c161s_1**

Acetyl-(acp) + 7 Malonyl-(acp) + 13 NADPH + 13 H(+) ⇄ Hexadecenoyl-(acp) + 13
NADP(+) + 7 CO(2) + 7 ACP + 7 H(2)O

\[ \text{NADP}(+) + 8 \text{Malonyl}(\text{acp}) + 16 \text{NADPH} + 16 \text{H} (+) \rightarrow \text{octadecanoyl}(\text{acp}) + 16 \text{NAD} (+) + 8 \text{CO}(2) + 8 \text{ACP} + 8 \text{H}(2)\text{O} \]

CDPk_1 \[ \text{ATP} + \text{CDP} \rightarrow \text{ADP} + \text{CTP} \]

CDPk_2 \[ \text{ATP} + \text{dCDP} \rightarrow \text{ADP} + \text{dCTP} \]

chem_1 \[ 2(\text{acetolactate} + \text{O}(2)) \rightarrow 2(\text{diacetyl} + 2 \text{CO}(2) + 2 \text{H}(2)\text{O}) \]

chem_2 \[ \text{L-glutamate 5-semialdehyde} \rightarrow 1(\text{pyrroline-5-carboxylate} + \text{H}(2)\text{O}) \]

chem_3 \[ \text{L-cysteine} \rightarrow \text{cysteamine} + \text{CO}(2) \]

d190s_1 \[ \text{octadecenoyl}(\text{acp}) \rightarrow 11,12(\text{methylene-octadecanoyl})(\text{acp}) \]

GDPk_2 \[ \text{dGDP} + \text{ATP} \rightarrow \text{dGTP} + \text{ADP} \]

trePP_1 \[ \text{trehalose 6-phosphate} + \text{phosphate} \rightarrow 6(\text{phospho-D-glucosamine} + \text{beta-D-glucose 1-phosphate}) \]

UDPk_1 \[ \text{ATP} + \text{UDP} \rightarrow \text{ADP} + \text{UTP} \]

unk_1 \[ \text{D-glucono-1,5-lactone} + \text{H}(2)\text{O} \rightarrow 6(\text{phospho-D-gluconate}) \]

unk_20 \[ \text{S-adenosyl-L-homocysteine} + \text{H}(2)\text{O} \rightarrow \text{adenosine} + \text{L-homocysteine} \]

unk_21 \[ \text{L-methionine} + \text{H}(2)\text{O} \rightarrow \text{methanethiol} + 2(\text{oxobutanoate} + \text{NH}(3)) \]

unk_23 \[ 4(\text{methylthio} 2(\text{oxobutyrate}) \rightarrow \text{methional} + \text{CO}(2) \]

unk_24 \[ \text{L-methionine} + 4(\text{methyl-2-oxopentanoate} \rightarrow 4(\text{methylthio} 2(\text{oxobutyrate} + \text{L-leucine} \rightarrow 4(\text{methylthio} 2(\text{oxobutyrate} + \text{L-}

unk_25 \[ \text{L-methionine} + 3(\text{methyl-2-oxopentanoate} \rightarrow 4(\text{methylthio} 2(\text{oxobutyrate} + \text{L-}

unk_26 \[ \text{L-methionine} + \text{phenylpyruvate} \rightarrow 4(\text{methylthio} 2(\text{oxobutyrate} + \text{L-phenylalanine} \]
L-methionine + 4-hydroxyphenylpyruvate <=> 4-methylthio 2-oxobutyrate + L-tyrosine

L-methionine + indol-3-pyruvate <=> 4-methylthio 2-oxobutyrate + L-tryptophan

LL-2,6-diaminoheptanedioate <=> meso-2,6-diaminoheptanedioate

4-hydroxyphenylpyruvate + 0.5 O(2) => 4-hydroxyphenylacetate + CO(2)

phenylacetaldehyde + NAD(+) + H(2)O => phenylacetate + NADH + H(+)

indol-3-acetaldehyde + NAD(+) + H(2)O => indol-3-acetate + NADH + H(+)

indol-3-pyruvate + NADH + H(+) => indol-3-lactate + NAD(+)

undecaprenyl diphosphate + H(2)O => undecaprenyl phosphate + phosphate

diacetylglycerol 3-phosphate + H(2)O => diacylglycerol + phosphate

lysophosphatidylglycerol 1-phosphate + H(2)O => lysophosphatidylglycerol + phosphate

UDP-glucose + diacylglycerol => UDP + monoglycosyl diacylglycerol

UDP-glucose + monoglycosyl diacylglycerol => UDP + diglycosyl diacylglycerol

2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine triphosphate + 3 H(2)O => 2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine + 3 phosphate

succinyl-CoA + ADP + phosphate <=> succinate + CoA + ATP

nicotinamide + D-ribose + ATP => nicotinamide ribonucleotide + H(2)O + ADP

NADP(+) + H(2)O => NAD(+) + phosphate

sulfate + ATP + GTP + H(2)O => diposphate + adenylyl sulfate + GDP + phosphate

3-phosphoadenylyl sulfate + oxidized thioredoxin => oxidized thioredoxin + H(2)SO(3) + adenosine 3',5'-bisphosphate

H(2)SO(3) + 3 NADPH + 3 H(+) => H(2)S + 3 NADP(+) + 3 H(2)O

(S)-malate + NAD(+) <=> oxaloacetate + NADH + H(+)

3MOBextI => 3MOBextX
3MOBextO => 3MOBextX
3MOPextI => 3MOPextX
3MOPextO => 3MOPextX
4MOPextI => 4MOPextX
4MOPextO => 4MOPextX
AALDextI => AALDextX
AALDextO => AALDextX
Acetyl-CoA => AcCoAextX
O

ACETextI  ACETextX => ACEText
ACETextO  ACEText => ACETextX
ACTNextI  ACTNextX => ACTNext
ACTNextO  ACTNext => ACTNextX
ADEextI  ADEextX => ADEext
ADEextO  ADEext => ADEextX
ALAextI  ALAextX => ALAext
ALAextO  ALAext => ALAextX
ALCTTextI  ALCTTextX => ALCTText
ALCTTextO  ALCTText => ALCTTextX
amtB_1  NH(3)ext <=> NH(3)
ARBTextI  ARBTextX => ARBText
ARBTextO  ARBText => ARBTextX
ARGextI  ARGextX => ARGext
ARGextO  ARGext => ARGextX
ASNextI  ASNextX => ASNext
ASNextO  ASNext => ASNextX
ASPextI  ASPextX => ASPext
ASPextO  ASPext => ASPextX
BDOHextI  BDOHextX => BDOHext
BDOHextO  BDOHext => BDOHextX
BIOTextI  BIOTextX => BIOText
BIOTextO  BIOText => BIOTextX
C140extI  C140extX => C140ext
C140extO  C140ext => C140extX
C160extI  C160extX => C160ext
C160extO  C160ext => C160extX
C180extI  C180extX => C180ext
C180extO  C180ext => C180extX
CELBextI  CELBextX => CELBext
CELBextO  CELBext => CELBextX
CHOextI  CHOextX => CHOext
CHOextO  CHOext => CHOextX
cholS_1  CHOext + ATP + H(2)O => choline + ADP + phosphate
GLUextO → GLUextX
GLUMextI → GLUMextX
GLUMextO → GLUMextX
GLYC3PextI → GLYC-3-PextX
GLYC3PextO → GLYC-3-PextX
GLYCextI → GLYCextX
GLYCextO → GLYCextX
GLYextI → GLYextX
GLYextO → GLYextX
GNNextI → GNNextX
GNNextO → GNNextX
HextI → H(+)-extX
HextO → H(+)-extX
H2OextI → H(2)OextX
H2OextO → H(2)OextX
HISextI → HISextX
HISextO → HISextX
HPAextI → HPAextX
HPAextO → HPAextX
HPLextI → HPLextX
HPLextO → HPLextX
HYXNextI → HYXNextX
HYXNextO → HYXNextX
I3AextI → I3AextX
I3AextO → I3AextX
I3LextI → I3LextX
I3LextO → I3LextX
ILEextI → ILEextX
ILEextO → ILEextX
LACTextI → LACTextX
LACTextO → LACTextX
LCTTextI → LCTTextX
LCTTextO → LCTTextX
LEUextI → LEUextX
LEUextO → LEUextX
<table>
<thead>
<tr>
<th>TPEPextO</th>
<th>TPEPext =&gt; TPEPextX</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRHLextI</td>
<td>TRHLextX =&gt; TRHLext</td>
</tr>
<tr>
<td>TRHLextO</td>
<td>TRHLext =&gt; TRHLextX</td>
</tr>
<tr>
<td>TRPextI</td>
<td>TRPextX =&gt; TRPext</td>
</tr>
<tr>
<td>TRPextO</td>
<td>TRPext =&gt; TRPextX</td>
</tr>
<tr>
<td>TYRextI</td>
<td>TYRextX =&gt; TYRext</td>
</tr>
<tr>
<td>TYRextO</td>
<td>TYRext =&gt; TYRextX</td>
</tr>
<tr>
<td>URAextI</td>
<td>URAextX =&gt; URAext</td>
</tr>
<tr>
<td>URAextO</td>
<td>URAext =&gt; URAextX</td>
</tr>
<tr>
<td>VALextI</td>
<td>VALextX =&gt; VALext</td>
</tr>
<tr>
<td>VALextO</td>
<td>VALext =&gt; VALextX</td>
</tr>
<tr>
<td>XANextI</td>
<td>XANextX =&gt; XANext</td>
</tr>
<tr>
<td>XANextO</td>
<td>XANext =&gt; XANextX</td>
</tr>
<tr>
<td>XYLANextI</td>
<td>XYLANextX =&gt; XYLANext</td>
</tr>
<tr>
<td>XYLANextO</td>
<td>XYLANext =&gt; XYLANextX</td>
</tr>
<tr>
<td>XYLextI</td>
<td>XYLextX =&gt; XYLext</td>
</tr>
<tr>
<td>XYLextO</td>
<td>XYLext =&gt; XYLextX</td>
</tr>
</tbody>
</table>

**Biomass**

\[
\begin{align*}
&4.201 \text{ PROT} + 0.074 \text{ DNA} + 0.329 \text{ RNA} + 0.064 \text{ POLYS} + 19 \text{ ATP} + 0.015 \text{ LTA} + 0.032 \text{ LIP} \\
&\rightarrow \text{ BIOMASS} + 19 \text{ ADP} + 19 \text{ phosphate}
\end{align*}
\]

**Intimin**

\[
\begin{align*}
&23 \text{ glycine} + 27 \text{ L-alanine} + 27 \text{ L-valine} + 13 \text{ L-leucine} + 16 \text{ L-isoleucine} + 30 \text{ L-serine} + 2 \text{ L-cysteine} + 34 \text{ L-threonine} + 1 \text{ L-methionine} + 6 \text{ L-proline} + 6 \text{ L-phenylalanine} + 11 \text{ L-tyrosine} + 4 \text{ L-tryptophan} + 29 \text{ L-lysine} + 2 \text{ L-arginine} + 14 \text{ L-aspartate} + 9 \text{ L-glutamate} + 21 \text{ L-asparagine} + 11 \text{ L-glutamine} + 1231.52 \text{ ATP} + 944.516 \text{ H(2)O} \\
&\rightarrow \text{ Intimin} + 1231.52 \text{ ADP} + 1231.52 \text{ phosphate}
\end{align*}
\]

**LinPass**

\[
\begin{align*}
&18.9 \text{ phosphatidylglycerol} + 42.5 \text{ cardiolipin} + 30.3 \text{ diglucosyl diacylglycerol} + 4 \text{ monoglucosyl diacylglycerol} + 4.3 \text{ lyposphatidylglycerol} \\
&\rightarrow 100 \text{ LIP}
\end{align*}
\]

**Lump_1**

\[
\begin{align*}
&15 \text{ phosphatidylglycerol} + \text{ glycerol 3-phosphate} \\
&\rightarrow 16 \text{ glycerol phosphate} + 15 \text{ diacylglycerol}
\end{align*}
\]

**Lump_2**

\[
\begin{align*}
&6.08 \text{ D-alanine} + 9.76 \text{ UDP-galactose} + 16 \text{ glycerol phosphate} \\
&\rightarrow \text{ D-alanyl-D-galactosyl-diphosphate} + 9.76 \text{ UDP} + 6.08 \text{ H(2)O}
\end{align*}
\]

**Lump_3**

\[
\begin{align*}
&\text{ D-alanyl-D-galactosyl-poly(glycerol phosphate)} + 9.76 \text{ UDP} \\
&\rightarrow \text{ D-alanine} + \text{ undecaprenyl diphosphate} + \text{ PG}
\end{align*}
\]

**Lump_4**

\[
\begin{align*}
&8.6 \text{ L-alanine} + 4.1 \text{ L-arginine} + 3.1 \text{ L-aspartate} + 5.9 \text{ L-asparagine} + 3.4 \text{ L-cysteine} + 3.6 \text{ L-glutamate} + 6.4 \text{ L-glutamine} + 9.2 \text{ glycine} + 1.5 \text{ L-histidine} + 6.1 \text{ L-isoleucine} + 8.7 \text{ L-leucine} + 7.2 \text{ L-lysine} + 2.5 \text{ L-methionine} + 3.8 \text{ L-phenylalanine} + 3.5 \text{ L-proline} + 5.1 \text{ L-serine} + 5.6 \text{ L-threonine} + 1.7 \text{ L-tryptophan} + 2.7 \text{ L-tyrosine} + 7.2 \text{ L-valine} + 430.6 \text{ ATP} \\
&\rightarrow 100 \text{ PROT} + 430.6 \text{ ADP} + 430.6 \text{ phosphate}
\end{align*}
\]

**Protass**

\[
\begin{align*}
&26.2 \text{ AMP} + 20 \text{ CMP} + 32.2 \text{ GMP} + 21.6 \text{ UMP} + 40 \text{ ATP} \\
&\rightarrow 100 \text{ RNA} + 40 \text{ ADP} + 40 \text{ phosphate} + 100 \text{ diphosphate}
\end{align*}
\]

**RNAass**

\[
\begin{align*}
&\text{ tripeptide} + 2 \text{ H(2)O} \\
&\rightarrow 0.233244 \text{ glycine} + 0.219169 \text{ L-alanine} + 0.17996 \text{ L-valine} + 0.165885 \text{ L-leucine} + 0.162869 \text{ L-isoleucine} + 0.0552949 \text{ L-serine} + 0.0482574 \text{ L-cysteine} + 0.0492627 \text{ L-threonine} + 0.036193 \text{ L-methionine} + 0.313673 \text{ L-proline}
\end{align*}
\]
glycerol 3-phosphate + 0.2941 Hexadecanoyl-(acp) + 0.0270 Hexadecenoyl-(acp) + 0.0116 octadecanoyl-(acp) + 0.4359 octadecenoyl-(acp) + 0.1341 11,12-methylene-octadecanoyl-(acp) => ACP + acylglycerol 3-phosphate

glycerol 3-phosphate + 0.1825 myristoyl-(acp) + 0.0121 tetradecenoyl-(acp) + 0.5883 Hexadecanoyl-(acp) + 0.0539 Hexadecenoyl-(acp) + 0.0231 octadecanoyl-(acp) + 0.8718 octadecenoyl-(acp) + 0.2682 11,12-methylene-octadecanoyl-(acp) => 2 ACP + diacylglycerol 3-phosphate

arcD1_1 ARGext + L-ornithine => L-arginine + ORNext

arcD2_1 ARGext + L-ornithine => L-arginine + ORNext

brnQ_1 LEUext + H(+)-ext <=> L-leucine + H(+)

brnQ_2 ILEext + H(+)-ext <=> L-isoleucine + H(+)

brnQ_3 VALExt + H(+)-ext <=> L-valine + H(+)

dipeptide + H(2)O => 0.155496 glycine + 0.146113 L-alanine + 0.119973 L-valine + 0.110597 L-leucine + 0.108579 L-isoleucine + 0.0368633 L-serine + 0.0321716 L-cysteine + 0.0328418 L-threonine + 0.0241287 L-methionine + 0.209115 L-proline + 0.0764075 L-phenylalanine + 0.0201072 L-tyrosine + 0.058311 L-tryptophan + 0.0542895 L-histidine + 0.066359 L-lysine + 0.0938338 L-arginine + 0.188338 L-aspartate + 0.394102 L-glutamate + 0.0603217 L-asparagine + 0.0120643 L-glutamine

dtpP_1 DPEPext + ATP + H(2)O => dipeptide + ADP + phosphate

dtpP_2 TPEPext + ATP + H(2)O => tripeptide + ADP + phosphate

dtpT_1 DPEPext + H(+)-ext <=> dipeptide + H(+)

dtpT_2 TPEPext + H(+)-ext <=> tripeptide + H(+)

E4PextO D-erythrose 4-phosphate => E4PextX

F6PextO D-fructose 6-phosphate => F6PextX

G3PextO D-glyceraldehyde 3-phosphate => G3PextX

G6PextO D-glucose 6-phosphate => G6PextX

gadC_1 GLUext + 4-aminobutanoate => L-glutamate + GABAext

glnP_1 GLNext + ATP => L-glutamine + ADP + phosphate

glpT_1 GLYC-3-Pext <=> glycerol 3-phosphate

glpP_1 GLUext + ATP => L-glutamate + ADP + phosphate

gleS_1 GLUext + ATP => L-glutamate + ADP + phosphate

gleS_2 ARGext + ATP => L-arginine + ADP + phosphate

gntP_1 GLCNext + H(+)-ext <=> D-glucurate + H(+)

GPextO glycerone phosphate => GPextX

H2O_trans_1 H(2)Oext <=> H(2)O

heme_1 2 NADH + 4 H(+) + O(2) => 2 NAD(+) + 2 H(+)ext + 2 H(2)O

lacS_1 LACText + H(+)ext <=> lactose + H(+)

220
lysP_1  LYS_ext + H(+) <- L-lysine + H(+)
lysQ_1  LYS_ext + H(+) <- L-lysine + H(+)
malF_1  MAL_ext + H(+) <- maltose + H(+)
malG_1  MAL_ext + H(+) <- maltose + H(+)
mlgP_1  MAL_ext <- (S)-malate
mtlA_1  GLUC_ext + phosphoenolpyruvate -> D-glucose 6-phosphate + Pyruvate
mtlA_2  MNTL_ext + phosphoenolpyruvate -> D-mannitol 1-phosphate + Pyruvate
nah_1  Na(+) + H(+) + ADP + phosphate -> Na(+) + H(+) + ATP + H(2)O
nah_2  Na(+) + H(+) + ATP + H(2)O -> Na(+) + H(+) + ADP + phosphate
OAextO  oxaloacetate -> OAextX
OGextO  2-oxoglutarate -> OGextX
Oppo  OPEP_ext + ATP -> oligopeptide + ADP + phosphate
p3MOB_1  3-methyl-2-oxobutanoate -> 3MOB_ext
p3MOP_1  3-methyl-2-oxopentanoate -> 3MOP_ext
p4MOP_1  4-methyl-2-oxopentanoate -> 4MOP_ext
pAAld_1  acetaldehyde <- AALD_ext
pACET_1  acetate + H(+) <- ACET_ext + H(+)ext
pACTN_1  2-acetoin <- ACTN_ext
pADE_1  ADE_ext + H(+) <- adenine + H(+)ext
pALA_1  ALA_ext + H(+) <- L-alanine + H(+)ext
pALCTT_1  2-acetolactate <- ALCTT_ext
pASN_1  ASN_ext + H(+) <- L-asparagine + H(+)ext
pASP_1  ASP_ext + H(+) <- L-aspartate + H(+)ext
pBDOH_1  (R,R)-butane-2,3-diol <- BDOH_ext
pBIOT_1  BIOT_ext + H(+)ext <- biotin + H(+)ext
pbuX_1  XAN_ext + H(+) <- xanthine + H(+)ext
pC14O_1  C14O_ext <- myristic
pC16O_1  C16O_ext <- palmitate
pC18O_1  C18O_ext <- stearate
pCO2_1  CO(2) <- CO(2)ext
pCYS_1  CYS_ext + H(+) <- L-cysteine + H(+)ext
pCYST_1  CYST_ext + H(+) <- cysteine + H(+)ext
pDTYL_1  diacetyl <- DTYL_ext
PEPextO  phosphoenolpyruvate -> PEPextX
pETOH_1  ethanol <-> ETOHext
pFOL_1   FOLext + H(+)ext -> folate + H(+)
pFORM_1  formate <-> FORMext
pGABA_1  4-aminobutanoate -> GABAext
pGALC_1  GALCext + H(+)ext <-> D-galactose + H(+)
pGLN     L-glutamine => GLNext
pGLUC_1  GLUCext + H(+)ext => D-glucose + H(+)
pGLY_1   GLYext + H(+)ext <-> glycine + H(+)
pGNN_1   GNNext + H(+)ext <-> guanine + H(+)
pHIS_1   HISext + H(+)ext <-> L-histidine + H(+)
pHPA_1   4-hydroxyphenylacetate-> HPAext
pHPL_1   4-hydroxyphenyllactate-> HPLext
pHYXN_1  HYXNext + H(+)ext <-> hypoxanthine + H(+)
pI3A_1   indol-3-acetate -> I3Aext
pI3L_1   indol-3-lactate -> I3Lext
pLCTT_1  (S)-lactate + H(+) <-> LCTText + H(+)ext
pMET_1   METext + H(+)ext <-> L-methionine + H(+)
pMTAL_1  methional -> MTALext
pMTTL_1  methanethiol -> MTTLext
pNICO_1  NICOext + H(+)ext => nicotinamide + H(+)
pO2_1    O2ext <-> O2
potB_1   SPRMext + ATP => spermidine + ADP + phosphate
potB_2   PUTRest + ATP => putrescine + ADP + phosphate
potC_1   SPRMext + ATP => spermidine + ADP + phosphate
potC_2   PUTRest + ATP => putrescine + ADP + phosphate
pPACT_1  phenylacetate -> PACText
pPAN_1   PANext + H(+)ext => pantothenate + H(+)
pPHE_1   PHEext + H(+)ext <-> L-phenylalanine + H(+)
pPLCT_1  phenyllactate -> PLCText
pPRO_1   L-proline <-> PROext
pPYR_1   Pyruvate + H(+) <-> PYRest + H(+)

pPYRO_1  PYROext + H(+)ext => pyrodal + H(+)
pRIBF_1  RIBFext + H(+)ext => riboflavin + H(+)
pSER_1   SERext + H(+)ext <-> L-serine + H(+)
pstC_1   Piext + H(+)ext => phosphate + H(+)
pstC_2   PPiext + H(+)ext => diphosphate + H(+)

Appendix 3a.

Mass spectral library used to identify metabolites in GC-MS analyzed samples.

<table>
<thead>
<tr>
<th>Name</th>
<th>RT</th>
<th>ref_ion1</th>
<th>ref_ion2</th>
<th>ref_ion3</th>
<th>ref_ion4</th>
<th>ion2to1</th>
<th>ion3to1</th>
<th>ion4to1</th>
</tr>
</thead>
<tbody>
<tr>
<td>d4-Alanine MCF derivative</td>
<td>10.947</td>
<td>106</td>
<td>62</td>
<td>74</td>
<td>107</td>
<td>0.163</td>
<td>0.114</td>
<td>0.041</td>
</tr>
<tr>
<td>Alanine MCF derivative</td>
<td>10.975</td>
<td>102</td>
<td>59</td>
<td>58</td>
<td>70</td>
<td>0.184</td>
<td>0.146</td>
<td>0.105</td>
</tr>
<tr>
<td>Methionine MCF derivative</td>
<td>18.087</td>
<td>115</td>
<td>61</td>
<td>147</td>
<td>114</td>
<td>0.947</td>
<td>0.824</td>
<td>0.372</td>
</tr>
<tr>
<td>Lysine MCF derivative</td>
<td>26.097</td>
<td>142</td>
<td>88</td>
<td>212</td>
<td>59</td>
<td>0.226</td>
<td>0.113</td>
<td>0.094</td>
</tr>
<tr>
<td>Glycine MCF derivative</td>
<td>11.386</td>
<td>88</td>
<td>44</td>
<td>59</td>
<td>56</td>
<td>0.206</td>
<td>0.182</td>
<td>0.117</td>
</tr>
<tr>
<td>Proline MCF derivative</td>
<td>14.89</td>
<td>128</td>
<td>42</td>
<td>82</td>
<td>129</td>
<td>0.138</td>
<td>0.102</td>
<td>0.07</td>
</tr>
<tr>
<td>Phenylalanine MCF derivative</td>
<td>19.8</td>
<td>162</td>
<td>91</td>
<td>146</td>
<td>178</td>
<td>0.762</td>
<td>0.436</td>
<td>0.23</td>
</tr>
<tr>
<td>Histidine MCF derivative</td>
<td>26.892</td>
<td>139</td>
<td>194</td>
<td>210</td>
<td>59</td>
<td>0.644</td>
<td>0.423</td>
<td>0.389</td>
</tr>
<tr>
<td>Threonine MCF derivative</td>
<td>15.554</td>
<td>115</td>
<td>100</td>
<td>147</td>
<td>59</td>
<td>0.22</td>
<td>0.208</td>
<td>0.147</td>
</tr>
<tr>
<td>Cysteine MCF derivative</td>
<td>19.873</td>
<td>192</td>
<td>176</td>
<td>59</td>
<td>132</td>
<td>0.861</td>
<td>0.619</td>
<td>0.522</td>
</tr>
<tr>
<td>Isoleucine MCF derivative</td>
<td>13.98</td>
<td>144</td>
<td>115</td>
<td>88</td>
<td>69</td>
<td>0.565</td>
<td>0.465</td>
<td>0.141</td>
</tr>
<tr>
<td>Aspartic acid MCF derivative</td>
<td>16.215</td>
<td>160</td>
<td>128</td>
<td>118</td>
<td>86</td>
<td>0.386</td>
<td>0.23</td>
<td>0.19</td>
</tr>
<tr>
<td>Glutamic acid MCF derivative</td>
<td>17.995</td>
<td>174</td>
<td>114</td>
<td>142</td>
<td>82</td>
<td>0.998</td>
<td>0.827</td>
<td>0.21</td>
</tr>
<tr>
<td>C10:0 acid MCF derivative (Capric acid)</td>
<td>12.418</td>
<td>74</td>
<td>87</td>
<td>143</td>
<td>43</td>
<td>0.578</td>
<td>0.241</td>
<td>0.168</td>
</tr>
<tr>
<td>2-Oxovaleric acid MCF derivative</td>
<td>7.167</td>
<td>71</td>
<td>43</td>
<td>41</td>
<td>59</td>
<td>0.905</td>
<td>0.335</td>
<td>0.114</td>
</tr>
<tr>
<td>Glyoxylic acid MCF derivative</td>
<td>11.085</td>
<td>75</td>
<td>103</td>
<td>59</td>
<td>47</td>
<td>0.647</td>
<td>0.332</td>
<td>0.3</td>
</tr>
<tr>
<td>2-Isopropylmalic acid MCF derivative</td>
<td>12.711</td>
<td>145</td>
<td>71</td>
<td>43</td>
<td>101</td>
<td>0.939</td>
<td>0.521</td>
<td>0.353</td>
</tr>
<tr>
<td>3-Hydroxydecanoic acid MCF derivative</td>
<td>15.206</td>
<td>103</td>
<td>74</td>
<td>71</td>
<td>43</td>
<td>0.266</td>
<td>0.255</td>
<td>0.239</td>
</tr>
<tr>
<td>C14:0 acid MCF derivative (Myristic acid)</td>
<td>17.602</td>
<td>74</td>
<td>87</td>
<td>143</td>
<td>43</td>
<td>0.681</td>
<td>0.227</td>
<td>0.189</td>
</tr>
<tr>
<td>Hippuric acid MCF derivative</td>
<td>21.216</td>
<td>105</td>
<td>77</td>
<td>134</td>
<td>161</td>
<td>0.344</td>
<td>0.222</td>
<td>0.09</td>
</tr>
<tr>
<td>Phthalic acid, monoester FROM PLASTIC TUBE</td>
<td>31.628</td>
<td>149</td>
<td>167</td>
<td>57</td>
<td>71</td>
<td>0.335</td>
<td>0.155</td>
<td>0.126</td>
</tr>
<tr>
<td>Levulinic acid MCF derivative</td>
<td>8.607</td>
<td>43</td>
<td>99</td>
<td>115</td>
<td>55</td>
<td>0.527</td>
<td>0.495</td>
<td>0.227</td>
</tr>
<tr>
<td>Itaconic acid MCF derivative</td>
<td>10.069</td>
<td>127</td>
<td>59</td>
<td>126</td>
<td>99</td>
<td>0.914</td>
<td>0.723</td>
<td>0.592</td>
</tr>
<tr>
<td>Citraconic acid MCF derivative</td>
<td>10.131</td>
<td>127</td>
<td>59</td>
<td>99</td>
<td>39</td>
<td>0.195</td>
<td>0.168</td>
<td>0.093</td>
</tr>
<tr>
<td>Nicotinic acid MCF derivative</td>
<td>10.696</td>
<td>106</td>
<td>78</td>
<td>137</td>
<td>136</td>
<td>0.703</td>
<td>0.637</td>
<td>0.318</td>
</tr>
<tr>
<td>p-Toluic acid MCF derivative</td>
<td>11.454</td>
<td>119</td>
<td>91</td>
<td>150</td>
<td>65</td>
<td>0.395</td>
<td>0.377</td>
<td>0.117</td>
</tr>
<tr>
<td>3-Hydroxybenzoic acid MCF derivative</td>
<td>17.018</td>
<td>135</td>
<td>210</td>
<td>107</td>
<td>179</td>
<td>0.321</td>
<td>0.226</td>
<td>0.199</td>
</tr>
<tr>
<td>Sebacic acid MCF derivative</td>
<td>17.612</td>
<td>74</td>
<td>125</td>
<td>199</td>
<td>55</td>
<td>0.964</td>
<td>0.934</td>
<td>0.931</td>
</tr>
<tr>
<td>Vanillic acid MCF derivative</td>
<td>20.709</td>
<td>165</td>
<td>240</td>
<td>196</td>
<td>181</td>
<td>0.628</td>
<td>0.453</td>
<td>0.202</td>
</tr>
<tr>
<td>Compound</td>
<td>4A</td>
<td>178</td>
<td>209</td>
<td>146</td>
<td>177</td>
<td>0.926</td>
<td>0.855</td>
<td>0.182</td>
</tr>
<tr>
<td>-------------------------------------------------------</td>
<td>----------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>4-Aminobenzoic acid MCF derivative</td>
<td>23.169</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sinapic acid MCF derivative</td>
<td>29.619</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Putrescine MCF derivative</td>
<td>21.528</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-Ornithine MCF derivative</td>
<td>24.667</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L-Norvaline MCF derivative</td>
<td>13.336</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EDTA MCF derivative</td>
<td>27.241</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid MCF derivative</td>
<td>7.689</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Succinic acid MCF derivative</td>
<td>8.955</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lactic acid MCF derivative</td>
<td>9.103</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-Hydroxyoctanoic acid MCF derivative</td>
<td>12.407</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Hydroxycinnamic acid MCF derivative</td>
<td>21.048</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Indole-3-butryic acid MCF derivative</td>
<td>25.964</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-Acetyl-L-cysteine MCF derivative</td>
<td>21.083</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-Phenylethanol</td>
<td>9.837</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Hydroxyphenylethanol MCF derivative</td>
<td>18.935</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oxalic acid MCF derivative</td>
<td>5.892</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-Aminocyclopropane-1-carboxylic acid MCF derivative</td>
<td>13.347</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,6-Diaminopimelic acid MCF derivative</td>
<td>29.397</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suberic acid MCF derivative</td>
<td>15.074</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tartaric acid MCF derivative</td>
<td>20.31</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-oxotetrahydrofuran-2-carboxylic acid MCF derivative</td>
<td>14.186</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Hydroxyphenilactic acid MCF derivative</td>
<td>18.827</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Terephthalic acid diester PLASTIC</td>
<td>15.973</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Anthranilic acid MCF derivative</td>
<td>14.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citramalic acid MCF derivative</td>
<td>10.578</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,4-Diaminobutyric acid MCF derivative</td>
<td>22.831</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid MCF derivative</td>
<td>7.787</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Aminoadipic acid MCF derivative</td>
<td>19.786</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Azelaic acid MCF derivative</td>
<td>16.363</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citric acid MCF derivative secondary peak</td>
<td>15.831</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Citric acid MCF derivative</td>
<td>16.229</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Substance</td>
<td>Peak Height</td>
<td>Width</td>
<td>Height</td>
<td>Averagel</td>
<td>SD</td>
<td>CV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
<td>-------</td>
<td>--------</td>
<td>----------</td>
<td>----</td>
<td>----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Hydroxyisobutyric acid MCF derivative</td>
<td>9.009</td>
<td>73</td>
<td>117</td>
<td>59</td>
<td>43</td>
<td>0.591</td>
<td>0.237</td>
<td>0.233</td>
</tr>
<tr>
<td>Isocitric acid MCF derivative secondary peak</td>
<td>20.257</td>
<td>129</td>
<td>59</td>
<td>157</td>
<td>201</td>
<td>0.529</td>
<td>0.443</td>
<td>0.334</td>
</tr>
<tr>
<td>Isocitric acid MCF derivative</td>
<td>20.605</td>
<td>129</td>
<td>59</td>
<td>157</td>
<td>101</td>
<td>0.538</td>
<td>0.451</td>
<td>0.357</td>
</tr>
<tr>
<td>2-Oxobutyric acid MCF derivative</td>
<td>5.702</td>
<td>57</td>
<td>59</td>
<td>116</td>
<td>56</td>
<td>0.155</td>
<td>0.106</td>
<td>0.05</td>
</tr>
<tr>
<td>Tetrachloroethane FROM CHLOROFORME</td>
<td>6.878</td>
<td>83</td>
<td>85</td>
<td>95</td>
<td>131</td>
<td>0.652</td>
<td>0.142</td>
<td>0.117</td>
</tr>
<tr>
<td>Pimelic acid MCF derivative</td>
<td>13.714</td>
<td>115</td>
<td>74</td>
<td>55</td>
<td>157</td>
<td>0.581</td>
<td>0.476</td>
<td>0.44</td>
</tr>
<tr>
<td>Quinic acid MCF derivative</td>
<td>16.315</td>
<td>191</td>
<td>206</td>
<td>192</td>
<td>57</td>
<td>0.162</td>
<td>0.143</td>
<td>0.133</td>
</tr>
<tr>
<td>C6:0 acid MCF derivative (Capronic acid)</td>
<td>6.185</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>99</td>
<td>0.409</td>
<td>0.315</td>
<td>0.277</td>
</tr>
<tr>
<td>Serine MCF derivative</td>
<td>17.256</td>
<td>100</td>
<td>59</td>
<td>176</td>
<td>42</td>
<td>0.253</td>
<td>0.149</td>
<td>0.138</td>
</tr>
<tr>
<td>4-Hydroxy-L-proline MCF derivative</td>
<td>19.733</td>
<td>144</td>
<td>126</td>
<td>82</td>
<td>59</td>
<td>0.117</td>
<td>0.112</td>
<td>0.092</td>
</tr>
<tr>
<td>2-Oxoglutaric acid MCF derivative</td>
<td>13.859</td>
<td>115</td>
<td>55</td>
<td>59</td>
<td>87</td>
<td>0.398</td>
<td>0.276</td>
<td>0.094</td>
</tr>
<tr>
<td>3,5-Diiodo-L-tyrosine MCF derivative</td>
<td>11.249</td>
<td>429</td>
<td>147</td>
<td>74</td>
<td>325</td>
<td>0.818</td>
<td>0.811</td>
<td>0.768</td>
</tr>
<tr>
<td>2-Methyloctadecanoic acid MCF derivative</td>
<td>24.269</td>
<td>88</td>
<td>101</td>
<td>43</td>
<td>55</td>
<td>0.412</td>
<td>0.157</td>
<td>0.147</td>
</tr>
<tr>
<td>Benzoic acid MCF derivative</td>
<td>9.492</td>
<td>105</td>
<td>77</td>
<td>136</td>
<td>51</td>
<td>0.62</td>
<td>0.39</td>
<td>0.231</td>
</tr>
<tr>
<td>C17:0 acid MCF derivative (Margaric acid)</td>
<td>22.976</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.74</td>
<td>0.295</td>
<td>0.263</td>
</tr>
<tr>
<td>C12:0 acid MCF derivative (Lauric acid)</td>
<td>15.085</td>
<td>74</td>
<td>87</td>
<td>55</td>
<td>41</td>
<td>0.649</td>
<td>0.234</td>
<td>0.224</td>
</tr>
<tr>
<td>C16:0 acid MCF derivative (Palmitic acid)</td>
<td>21.064</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.718</td>
<td>0.266</td>
<td>0.241</td>
</tr>
<tr>
<td>C18:0 acid MCF derivative (Stearic acid)</td>
<td>24.007</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.739</td>
<td>0.299</td>
<td>0.256</td>
</tr>
<tr>
<td>beta-Alanine MCF derivative</td>
<td>12.652</td>
<td>88</td>
<td>101</td>
<td>102</td>
<td>130</td>
<td>0.927</td>
<td>0.379</td>
<td>0.352</td>
</tr>
<tr>
<td>N2-Acetyl-L-lysine MCF derivative</td>
<td>28.903</td>
<td>129</td>
<td>57</td>
<td>112</td>
<td>70</td>
<td>0.311</td>
<td>0.281</td>
<td>0.268</td>
</tr>
<tr>
<td>Pyroglutamic acid MCF derivative</td>
<td>16.427</td>
<td>84</td>
<td>41</td>
<td>56</td>
<td>143</td>
<td>0.15</td>
<td>0.083</td>
<td>0.057</td>
</tr>
<tr>
<td>2-Hydroxybutyric acid MCF derivative</td>
<td>10.3</td>
<td>117</td>
<td>73</td>
<td>59</td>
<td>45</td>
<td>0.964</td>
<td>0.89</td>
<td>0.463</td>
</tr>
<tr>
<td>Tyrosine MCF derivative</td>
<td>28.558</td>
<td>121</td>
<td>236</td>
<td>165</td>
<td>161</td>
<td>0.994</td>
<td>0.53</td>
<td>0.242</td>
</tr>
<tr>
<td>5-Methoxytryptophane MCF derivative</td>
<td>39.918</td>
<td>160</td>
<td>161</td>
<td>145</td>
<td>306</td>
<td>0.112</td>
<td>0.089</td>
<td>0.068</td>
</tr>
<tr>
<td>Salicylic acid MCF derivative</td>
<td>16.798</td>
<td>135</td>
<td>133</td>
<td>121</td>
<td>92</td>
<td>0.491</td>
<td>0.413</td>
<td>0.41</td>
</tr>
<tr>
<td>2-Phosphoenolpyruvic acid MCF derivative</td>
<td>14.474</td>
<td>109</td>
<td>127</td>
<td>79</td>
<td>179</td>
<td>0.572</td>
<td>0.147</td>
<td>0.136</td>
</tr>
<tr>
<td>Homocysteine MCF derivative</td>
<td>23.193</td>
<td>114</td>
<td>174</td>
<td>59</td>
<td>146</td>
<td>0.851</td>
<td>0.378</td>
<td>0.275</td>
</tr>
<tr>
<td>5-Hydroxy-L-lysine MCF derivative</td>
<td>30.085</td>
<td>101</td>
<td>88</td>
<td>141</td>
<td>56</td>
<td>0.546</td>
<td>0.274</td>
<td>0.24</td>
</tr>
<tr>
<td>Caffeine</td>
<td>23.494</td>
<td>194</td>
<td>109</td>
<td>67</td>
<td>55</td>
<td>0.394</td>
<td>0.197</td>
<td>0.18</td>
</tr>
<tr>
<td>Valine MCF derivative</td>
<td>12.69</td>
<td>130</td>
<td>98</td>
<td>115</td>
<td>59</td>
<td>0.219</td>
<td>0.198</td>
<td>0.123</td>
</tr>
<tr>
<td>DBP</td>
<td>23.532</td>
<td>149</td>
<td>150</td>
<td>223</td>
<td>41</td>
<td>0.089</td>
<td>0.051</td>
<td>0.048</td>
</tr>
<tr>
<td>Compound</td>
<td>Mol. Wt</td>
<td>MCF</td>
<td>MCF</td>
<td>MCF</td>
<td>MCF</td>
<td>MCF</td>
<td>MCF</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>----------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>Benzothiazole</td>
<td>11.952</td>
<td>135</td>
<td>108</td>
<td>69</td>
<td>136</td>
<td>0.293</td>
<td>0.124</td>
<td>0.088</td>
</tr>
<tr>
<td>L-Cystathionine MCF derivative</td>
<td>34.577</td>
<td>160</td>
<td>114</td>
<td>59</td>
<td>115</td>
<td>0.718</td>
<td>0.378</td>
<td>0.371</td>
</tr>
<tr>
<td>Tryptophan MCF derivative</td>
<td>33.123</td>
<td>130</td>
<td>131</td>
<td>276</td>
<td>77</td>
<td>0.105</td>
<td>0.074</td>
<td>0.039</td>
</tr>
<tr>
<td>N-Acetyl-glutamic acid MCF derivative</td>
<td>18.961</td>
<td>84</td>
<td>116</td>
<td>158</td>
<td>43</td>
<td>0.83</td>
<td>0.69</td>
<td>0.368</td>
</tr>
<tr>
<td>Phenethyl acetate</td>
<td>12.47</td>
<td>104</td>
<td>43</td>
<td>91</td>
<td>105</td>
<td>0.528</td>
<td>0.212</td>
<td>0.111</td>
</tr>
<tr>
<td>Malonic acid MCF derivative</td>
<td>7.362</td>
<td>101</td>
<td>59</td>
<td>74</td>
<td>42</td>
<td>0.672</td>
<td>0.426</td>
<td>0.241</td>
</tr>
<tr>
<td>Leucine MCF derivative</td>
<td>14.044</td>
<td>144</td>
<td>88</td>
<td>102</td>
<td>59</td>
<td>0.808</td>
<td>0.187</td>
<td>0.158</td>
</tr>
<tr>
<td>trans-Cinnamic acid MCF derivative</td>
<td>14.277</td>
<td>131</td>
<td>103</td>
<td>162</td>
<td>77</td>
<td>0.605</td>
<td>0.533</td>
<td>0.344</td>
</tr>
<tr>
<td>cis-Aconitic acid MCF derivative</td>
<td>15.392</td>
<td>153</td>
<td>184</td>
<td>156</td>
<td>59</td>
<td>0.79</td>
<td>0.714</td>
<td>0.502</td>
</tr>
<tr>
<td>CB:0 acid MCF derivative (Caprylic acid)</td>
<td>9.354</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.44</td>
<td>0.216</td>
<td>0.186</td>
</tr>
<tr>
<td>C11:0 acid MCF derivative (Undecanoic acid)</td>
<td>13.788</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.582</td>
<td>0.209</td>
<td>0.196</td>
</tr>
<tr>
<td>C13:0 acid MCF derivative (Tridecanoic acid)</td>
<td>16.367</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>55</td>
<td>0.65</td>
<td>0.223</td>
<td>0.207</td>
</tr>
<tr>
<td>C15:0 acid MCF derivative (Pentadecanoic acid)</td>
<td>19.1</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>143</td>
<td>0.684</td>
<td>0.247</td>
<td>0.185</td>
</tr>
<tr>
<td>C16:1n-7 acid MCF derivative (Palmitoleic acid)</td>
<td>20.846</td>
<td>55</td>
<td>69</td>
<td>41</td>
<td>74</td>
<td>0.7</td>
<td>0.654</td>
<td>0.651</td>
</tr>
<tr>
<td>C18:1n-9 acid MCF derivative (Oleic acid)</td>
<td>23.89</td>
<td>55</td>
<td>69</td>
<td>74</td>
<td>41</td>
<td>0.72</td>
<td>0.661</td>
<td>0.607</td>
</tr>
<tr>
<td>C18:2n-6,9 acid MCF derivative (Linoleic acid)</td>
<td>23.938</td>
<td>67</td>
<td>81</td>
<td>55</td>
<td>95</td>
<td>0.896</td>
<td>0.784</td>
<td>0.638</td>
</tr>
<tr>
<td>C18:3n-6,9,12 acid MCF derivative (gamma-Linolenic acid)</td>
<td>24.124</td>
<td>79</td>
<td>67</td>
<td>95</td>
<td>93</td>
<td>0.602</td>
<td>0.534</td>
<td>0.488</td>
</tr>
<tr>
<td>C20:1n-9 acid MCF derivative (cis-Gondoic acid)</td>
<td>26.517</td>
<td>55</td>
<td>79</td>
<td>41</td>
<td>67</td>
<td>0.879</td>
<td>0.766</td>
<td>0.686</td>
</tr>
<tr>
<td>C20:2n-6,9 acid MCF derivative (cis-11,14-Eicosadienoic acid)</td>
<td>26.607</td>
<td>67</td>
<td>81</td>
<td>95</td>
<td>55</td>
<td>0.924</td>
<td>0.71</td>
<td>0.644</td>
</tr>
<tr>
<td>C20:0 acid MCF derivative (Arachidic acid)</td>
<td>26.808</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>326</td>
<td>0.752</td>
<td>0.334</td>
<td>0.272</td>
</tr>
<tr>
<td>C21:0 acid MCF derivative (Heneicosanoic acid)</td>
<td>28.475</td>
<td>74</td>
<td>87</td>
<td>43</td>
<td>340</td>
<td>0.763</td>
<td>0.356</td>
<td>0.307</td>
</tr>
<tr>
<td>C22:6n-3,6,9,12,15,18 acid MCF derivative (DHA)</td>
<td>29.419</td>
<td>79</td>
<td>91</td>
<td>67</td>
<td>93</td>
<td>0.851</td>
<td>0.552</td>
<td>0.447</td>
</tr>
<tr>
<td>C22:0 acid MCF derivative (Behenic acid)</td>
<td>29.768</td>
<td>74</td>
<td>87</td>
<td>354</td>
<td>43</td>
<td>0.781</td>
<td>0.371</td>
<td>0.37</td>
</tr>
<tr>
<td>C23:0 acid MCF derivative (Tricosanoic acid)</td>
<td>31.273</td>
<td>74</td>
<td>87</td>
<td>368</td>
<td>43</td>
<td>0.767</td>
<td>0.467</td>
<td>0.454</td>
</tr>
<tr>
<td>C24:1n-9 acid MCF derivative (Nervonic acid)</td>
<td>32.855</td>
<td>55</td>
<td>69</td>
<td>83</td>
<td>348</td>
<td>0.765</td>
<td>0.628</td>
<td>0.607</td>
</tr>
<tr>
<td>C24:0 acid MCF derivative (Lignoceric acid)</td>
<td>33.18</td>
<td>74</td>
<td>87</td>
<td>382</td>
<td>43</td>
<td>0.795</td>
<td>0.451</td>
<td>0.405</td>
</tr>
<tr>
<td>Chemical Formula</td>
<td>R</td>
<td>M</td>
<td>D</td>
<td>Y</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>C14:1n-5 acid MCF derivative</td>
<td>17.567</td>
<td>55</td>
<td>96</td>
<td>84</td>
<td>67</td>
<td>0.577</td>
<td>0.549</td>
<td>0.485</td>
</tr>
<tr>
<td>Myristoleic acid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C15:1n-5 acid MCF derivative</td>
<td>19.071</td>
<td>55</td>
<td>69</td>
<td>41</td>
<td>83</td>
<td>0.495</td>
<td>0.489</td>
<td>0.356</td>
</tr>
<tr>
<td>(cis-10-Pentadecenoic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C20:4n-6,9,12,15 acid MCF derivative</td>
<td>26.24</td>
<td>79</td>
<td>91</td>
<td>80</td>
<td>67</td>
<td>0.758</td>
<td>0.679</td>
<td>0.666</td>
</tr>
<tr>
<td>Arachidonic acid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C20:3n-6,9,12 acid MCF derivative</td>
<td>26.447</td>
<td>79</td>
<td>67</td>
<td>80</td>
<td>81</td>
<td>0.947</td>
<td>0.827</td>
<td>0.692</td>
</tr>
<tr>
<td>(bishomo-gamma-linolenic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C17:1n-7 acid MCF derivative</td>
<td>22.86</td>
<td>55</td>
<td>69</td>
<td>41</td>
<td>74</td>
<td>0.67</td>
<td>0.59</td>
<td>0.588</td>
</tr>
<tr>
<td>(cis-11,14,17-Eicosatrienoic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C22:1n-9 acid MCF derivative</td>
<td>29.544</td>
<td>55</td>
<td>69</td>
<td>83</td>
<td>97</td>
<td>0.753</td>
<td>0.603</td>
<td>0.548</td>
</tr>
<tr>
<td>Erucic acid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C22:2n-6,9 acid MCF derivative</td>
<td>29.634</td>
<td>67</td>
<td>81</td>
<td>95</td>
<td>55</td>
<td>0.94</td>
<td>0.737</td>
<td>0.708</td>
</tr>
<tr>
<td>(13,16-Docosadienoic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Malic acid MCF derivative</td>
<td>11.271</td>
<td>103</td>
<td>71</td>
<td>43</td>
<td>61</td>
<td>0.704</td>
<td>0.383</td>
<td>0.293</td>
</tr>
<tr>
<td>Dodecane</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tridecane</td>
<td>9.575</td>
<td>57</td>
<td>43</td>
<td>71</td>
<td>41</td>
<td>0.699</td>
<td>0.632</td>
<td>0.428</td>
</tr>
<tr>
<td>Pentadecane</td>
<td>11.124</td>
<td>57</td>
<td>71</td>
<td>43</td>
<td>85</td>
<td>0.685</td>
<td>0.678</td>
<td>0.452</td>
</tr>
<tr>
<td>Heptadecane</td>
<td>13.956</td>
<td>57</td>
<td>71</td>
<td>43</td>
<td>85</td>
<td>0.732</td>
<td>0.658</td>
<td>0.518</td>
</tr>
<tr>
<td>Tricosane</td>
<td>16.5</td>
<td>57</td>
<td>71</td>
<td>43</td>
<td>85</td>
<td>0.765</td>
<td>0.648</td>
<td>0.558</td>
</tr>
<tr>
<td>Nonacosane</td>
<td>25.344</td>
<td>57</td>
<td>71</td>
<td>43</td>
<td>85</td>
<td>0.787</td>
<td>0.648</td>
<td>0.606</td>
</tr>
<tr>
<td>gamma-Aminobutyric acid MCF derivative</td>
<td>35.406</td>
<td>57</td>
<td>71</td>
<td>43</td>
<td>85</td>
<td>0.755</td>
<td>0.638</td>
<td>0.572</td>
</tr>
<tr>
<td>Malonic acid D-MCF derivative</td>
<td>14.845</td>
<td>102</td>
<td>88</td>
<td>112</td>
<td>59</td>
<td>0.672</td>
<td>0.417</td>
<td>0.394</td>
</tr>
<tr>
<td>Succinic acid D-MCF derivative</td>
<td>27.499</td>
<td>79</td>
<td>91</td>
<td>67</td>
<td>93</td>
<td>0.785</td>
<td>0.549</td>
<td>0.475</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid MCF derivative</td>
<td>6.439</td>
<td>43</td>
<td>74</td>
<td>45</td>
<td>71</td>
<td>0.67</td>
<td>0.563</td>
<td>0.372</td>
</tr>
<tr>
<td>Malonic acid D-MCF derivative</td>
<td>7.22</td>
<td>104</td>
<td>62</td>
<td>78</td>
<td>42</td>
<td>0.945</td>
<td>0.466</td>
<td>0.277</td>
</tr>
<tr>
<td>Fumaric acid D-MCF derivative</td>
<td>8.845</td>
<td>118</td>
<td>55</td>
<td>62</td>
<td>117</td>
<td>0.599</td>
<td>0.434</td>
<td>0.334</td>
</tr>
<tr>
<td>Itaconic acid D-MCF derivative</td>
<td>8.98</td>
<td>116</td>
<td>62</td>
<td>88</td>
<td>54</td>
<td>0.2</td>
<td>0.132</td>
<td>0.054</td>
</tr>
<tr>
<td>Citramalic acid D-MCF derivative</td>
<td>9.988</td>
<td>62</td>
<td>130</td>
<td>129</td>
<td>102</td>
<td>0.667</td>
<td>0.556</td>
<td>0.422</td>
</tr>
<tr>
<td>d4-Alanine D-MCF derivative</td>
<td>10.506</td>
<td>128</td>
<td>84</td>
<td>146</td>
<td>145</td>
<td>0.299</td>
<td>0.284</td>
<td>0.25</td>
</tr>
<tr>
<td>Alanine D-MCF derivative</td>
<td>10.83</td>
<td>109</td>
<td>65</td>
<td>62</td>
<td>46</td>
<td>0.254</td>
<td>0.149</td>
<td>0.088</td>
</tr>
<tr>
<td>Malic acid D-MCF derivative</td>
<td>10.866</td>
<td>105</td>
<td>61</td>
<td>62</td>
<td>42</td>
<td>0.237</td>
<td>0.209</td>
<td>0.086</td>
</tr>
<tr>
<td>Valine D-MCF derivative</td>
<td>11.211</td>
<td>106</td>
<td>71</td>
<td>43</td>
<td>64</td>
<td>0.822</td>
<td>0.438</td>
<td>0.346</td>
</tr>
<tr>
<td>Isoleucine D-MCF derivative</td>
<td>13.824</td>
<td>147</td>
<td>91</td>
<td>105</td>
<td>62</td>
<td>0.703</td>
<td>0.196</td>
<td>0.174</td>
</tr>
<tr>
<td>Leucine D-MCF derivative</td>
<td>13.88</td>
<td>147</td>
<td>118</td>
<td>91</td>
<td>62</td>
<td>0.56</td>
<td>0.494</td>
<td>0.199</td>
</tr>
<tr>
<td>gamma-Aminobutyric acid D-MCF</td>
<td>14.626</td>
<td>105</td>
<td>91</td>
<td>47</td>
<td>62</td>
<td>0.715</td>
<td>0.586</td>
<td>0.487</td>
</tr>
</tbody>
</table>

228
<table>
<thead>
<tr>
<th>Compound</th>
<th>Amount</th>
<th>Retention Time</th>
<th>Rf</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proline D-MCF derivative</td>
<td>14.812</td>
<td>131</td>
<td>45</td>
<td>85</td>
</tr>
<tr>
<td>Threonine D-MCF derivative</td>
<td>15.403</td>
<td>118</td>
<td>62</td>
<td>153</td>
</tr>
<tr>
<td>Aspartic acid D-MCF derivative</td>
<td>16.068</td>
<td>166</td>
<td>131</td>
<td>124</td>
</tr>
<tr>
<td>Citric acid D-MCF derivative</td>
<td>16.104</td>
<td>146</td>
<td>104</td>
<td>62</td>
</tr>
<tr>
<td>Serine D-MCF derivative</td>
<td>17.11</td>
<td>103</td>
<td>62</td>
<td>182</td>
</tr>
<tr>
<td>Glutamic acid D-MCF derivative</td>
<td>17.813</td>
<td>117</td>
<td>180</td>
<td>145</td>
</tr>
<tr>
<td>Phenylalanine D-MCF derivative</td>
<td>19.555</td>
<td>165</td>
<td>91</td>
<td>45</td>
</tr>
<tr>
<td>Tartaric acid D-MCF derivative</td>
<td>20.193</td>
<td>62</td>
<td>88</td>
<td>114</td>
</tr>
<tr>
<td>cis-Aconitic acid D-MCF derivative</td>
<td>15.262</td>
<td>156</td>
<td>190</td>
<td>162</td>
</tr>
<tr>
<td>Oxalic acid D-MCF derivative</td>
<td>5.819</td>
<td>62</td>
<td>50</td>
<td>124</td>
</tr>
<tr>
<td>2-Oxovaleric acid D-MCF derivative</td>
<td>7.045</td>
<td>43</td>
<td>71</td>
<td>41</td>
</tr>
<tr>
<td>2,3-Butanediol</td>
<td>5.626</td>
<td>45</td>
<td>57</td>
<td>43</td>
</tr>
<tr>
<td>Fumaric acid MCF derivative</td>
<td>8.299</td>
<td>113</td>
<td>85</td>
<td>114</td>
</tr>
<tr>
<td>Lactic acid D-MCF derivative</td>
<td>8.8</td>
<td>62</td>
<td>106</td>
<td>43</td>
</tr>
<tr>
<td>C8:0 acid D-MCF derivative (Caprylic acid)</td>
<td>9.123</td>
<td>77</td>
<td>90</td>
<td>55</td>
</tr>
<tr>
<td>Glycine D-MCF derivative</td>
<td>11.114</td>
<td>91</td>
<td>47</td>
<td>92</td>
</tr>
<tr>
<td>C10:0 acid D-MCF derivative (Capric acid)</td>
<td>12.146</td>
<td>77</td>
<td>90</td>
<td>146</td>
</tr>
<tr>
<td>5-oxotetrahydrofuran-2-carboxylic acid D-MCF derivative</td>
<td>13.867</td>
<td>85</td>
<td>57</td>
<td>62</td>
</tr>
<tr>
<td>Citric acid D-MCF derivative secondary peak</td>
<td>15.543</td>
<td>104</td>
<td>146</td>
<td>44</td>
</tr>
<tr>
<td>Isocitric acid D-MCF derivative</td>
<td>20.243</td>
<td>135</td>
<td>62</td>
<td>107</td>
</tr>
<tr>
<td>C16:0 acid D-MCF derivative (Palmitic acid)</td>
<td>20.916</td>
<td>77</td>
<td>90</td>
<td>78</td>
</tr>
<tr>
<td>C18:0 acid D-MCF derivative (Stearic acid)</td>
<td>23.909</td>
<td>77</td>
<td>90</td>
<td>43</td>
</tr>
<tr>
<td>L-Ornithine D-MCF derivative</td>
<td>24.314</td>
<td>131</td>
<td>70</td>
<td>62</td>
</tr>
<tr>
<td>Lysine D-MCF derivative</td>
<td>25.728</td>
<td>145</td>
<td>91</td>
<td>146</td>
</tr>
<tr>
<td>Tyrosine D-MCF derivative</td>
<td>28.25</td>
<td>124</td>
<td>242</td>
<td>168</td>
</tr>
<tr>
<td>C22:4n-6,9,12,15 acid MCF derivative (Adrenic Acid)</td>
<td>29.357</td>
<td>79</td>
<td>80</td>
<td>67</td>
</tr>
<tr>
<td>C22:5n-3,6,9,12,15 acid MCF derivative (DPA)</td>
<td>29.595</td>
<td>79</td>
<td>91</td>
<td>67</td>
</tr>
<tr>
<td>C18:3n-3,6,9 acid MCF derivative (Linolenic acid)</td>
<td>23.721</td>
<td>79</td>
<td>67</td>
<td>80</td>
</tr>
<tr>
<td>Putrescine D-MCF derivative</td>
<td>21.349</td>
<td>91</td>
<td>47</td>
<td>131</td>
</tr>
</tbody>
</table>
Glutathione MCF derivative  19.493  142  98  42  41  0.268  0.168  0.16

NADH(NAD) MCF derivative  11.372  140  138  171  80  0.646  0.63  0.617

Appendix 3b.

R function buildLib.R for MacOS. This function converts an AMDIS library into the format required by MetaBox. See section 3.4.2.

```r
buildLib <- function(AmdisLib = tk_choose.files(multi=FALSE), folder = tk_choose.dir(), save = TRUE, output = "ion_lib”){
    #### Requires gtools
    library(gtools)
    if (is.data.frame(AmdisLib) == TRUE) {
        lib <- AmdisLib
        libOrig <- lib
    } else {
        lib <- data.frame(read.csv(AmdisLib,sep="t", header=FALSE))
        libOrig <- lib
    }
    if (save == TRUE){
        confirm_folder <- data.frame(file.info(folder))
        if (confirm_folder$isdir != TRUE){
            stop("folder is not an existing directory.")
        }
    }
    bar <- 0
toBar <- nrow(lib)
pb <- txtProgressBar(min = 0, max = toBar, style = 3, width = 50)
i <- 1
    ## Get the row containing the name of the compound ##
    getname <- c()
    while(length(getname) == 0){
        getname <- grep("NAME:", lib[i,], ignore.case=FALSE)
        i <- i + 1
    }
    ## Keep just the name of the compound ##
    getname <- grep("NAME:", lib[i-1,], ignore.case=FALSE)
    nameLoc <- i - 1
    ## Get Retention time ##
    getRT <- c()
    while(length(getRT) == 0){
        getRT <- grep("RT:", lib[i,], ignore.case=FALSE)
        i <- i + 1
    }
    ## Keep just the name of the compound ##
    getRT <- grep("RT:", lib[i-1,], ignore.case=FALSE)
    ## Get rows containing ions and intensities ##
    getion <- c()
    while(length(getion) == 0){
        getion <- grep("NUM PEAKS:", lib[i,], ignore.case=FALSE)
        i <- i + 1
    }
    startpeak <- i
    getname2 <- c()
    checkNames <- grep("NAME:", lib[i], ignore.case=FALSE)
    ```
if (length(checkNames) > 1) {
  while (length(getname2) == 0) {
    getname2 <- grep("NAME:", lib[,], ignore.case = FALSE)
    i <- i + 1
  }
  endpeak <- i - 2
} else {
  endpeak <- nrow(lib)
}

peak_rows <- lib[startpeak:endpeak,]
peak_rows <- data.frame(as.character(peak_rows))
for (k in 1:nrow(peak_rows)) {
  row1 <- data.frame(unlist(strsplit(as.character(peak_rows[k,]), "\(")))
  row1 <- data.frame(gsub("\)", "", row1[,1], fixed = TRUE))
  row1 <- data.frame(gsub("\", "", row1[,1], fixed = TRUE))
  row1 <- apply(row1[1], 1, function(x) data.frame(unlist(strsplit(x, "\\[[:blank:]\\]"))))
  row1 <- lapply(row1, function(x) x[x != ""])
  row1 <- unlist(row1)

  frags <- row1[odd(1:length(row1))]
  int <- row1[odd(1:length(row1))]
  oneGroup <- data.frame(cbind(frags, int))
  if (!exists("totalPeak")) {
    totalPeak <- oneGroup
  } else {
    totalPeak <- rbind(totalPeak, oneGroup)
  }
}

onePeak[1] <- as.numeric(as.character(totalPeak[1,1]))
onePeak[2] <- as.numeric(as.character(totalPeak[2]))
ions <- data.frame(totalPeak[2] == max(totalPeak[2], na.rm = TRUE),]
totalPeak <- totalPeak[-as.numeric(row.names(ions)),]
while (nrow(ions) < 4) {
  ions <- rbind(ions, totalPeak[totalPeak[2] == max(totalPeak[2], na.rm = TRUE),]}
toDelete <- totalPeak[totalPeak[2] == max(totalPeak[2], na.rm = TRUE),]
totalPeak <- totalPeak[-as.numeric(row.names(toDelete) != row.names(totalPeak),]}
}
rm(totalPeak)

### Now, prepare the data frame with name of compounds, ref_ion1, ref_ion2, etc...
if (!exists("finalLib")) {
  finalLib <- data.frame(Name = getname, RT = getRT, ref_ion1 = ions[1,1], ref_ion2 = ions[2,1], ref_ion3 = ions[3,1], ref_ion4 = ions[4,1], ion2to1 = as.numeric(ions[2,2])/1000, ion3to1 = as.numeric(ions[3,2])/1000, ion4to1 = as.numeric(ions[4,2])/1000)
} else {
  finalLib2 <- data.frame(Name = getname, RT = getRT, ref_ion1 = ions[1,1], ref_ion2 = ions[2,1], ref_ion3 = ions[3,1], ref_ion4 = ions[4,1], ion2to1 = as.numeric(ions[2,2])/1000, ion3to1 = as.numeric(ions[3,2])/1000, ion4to1 = as.numeric(ions[4,2])/1000)
  finalLib <- rbind(finalLib, finalLib2)
}

lib <- data.frame(lib[-(c(nameLoc:endpeak))])
bar <- bar + lengthc(nameLoc:endpeak))
Sys.sleep(0.0000001)
setTxtProgressBar(pb, bar)
}
if (save == TRUE) {
  sheet <- output
  store <- paste(folder, ",", sheet, ",.csv", sep = ""
  write.csv(finalLib, file = store, row.names = FALSE)
  print("Complete!")
  print(paste("The file ", output, ",.csv", " was saved in the folder ", folder, sep = ""))
} else {
  print("No file was saved because the argument save was set as FALSE")
}
return(finalLib)

############################################################### THE END ###################################################################
Appendix 3c.

R function `identify.R` for MacOS. This function is used to deconvolute and identify metabolites analyzed by GC-MS. See section 3.4.2.

```r
identify <- function(main.folder = tk_choose.dir(default = "Select the main folder where you stored the CDF files"),
                    ion.lib = read.csv(tk_choose.files(default = "Select the CSV file containing the ion library", multi = FALSE, caption = "Select the CSV file to be used as ion library.", sep = ","),
                    excludeLib = c("D-MCF"),
                    align = TRUE,
                    internal.standard = "d4-Alanine MCF derivative",
                    matchFactor = 0.5,
                    correlation = 0.95,
                    scoreCut = 11,
                    filterNoise = TRUE,
                    save = TRUE,
                    output = "MetaBox_data")
{
    ### require(splines, plyr, xcms, simecol)
    old.wd <- getwd()
    main.folder <- main.folder
    setwd(main.folder)
    files <- c(dir())
    info <- file.info(files)
    isDir <- info$isdir
    conditions <- c(files[isDir == TRUE])
    largest <- 1
    for (i in 1:length(conditions)) {
        largest <- c(largest, length(grep(".CDF", c(list.files(conditions[i], full.names = TRUE)), ignore.case = TRUE)))
    }
    largest <- max(largest)
    num.rep <- length(grep(".CDF", c(list.files(conditions[1], full.names = TRUE)), ignore.case = TRUE))
    samples <- grep(".CDF", c(list.files(conditions[1], full.names = TRUE)), ignore.case = TRUE, value = TRUE)
    replicates <- matrix(c(1:length(samples)), nrow = 1, ncol = largest)
    for (i in 2:length(conditions)) {
        num.rep <- c(num.rep, length(grep(".CDF", c(list.files(conditions[i], full.names = TRUE)), ignore.case = TRUE)))
        samples <- c(samples, grep(".CDF", c(list.files(conditions[i], full.names = TRUE)), ignore.case = TRUE, value = TRUE))
        up.num <- length(grep(".CDF", c(list.files(conditions[i], full.names = TRUE)), ignore.case = TRUE, value = TRUE))
        if (i < 3) {
            low.num <- (replicates[length(replicates)] + 1)
        } else {
            low.num <- max(replicates[1 - 1, !duplicated(replicates[1 - 1,])]) + 1
        }
        replicates <- rbind(replicates, c((low.num):(low.num + (up.num - 1))))
    }
    if (is.data.frame(ion.lib) == FALSE) {
        ion.lib = read.csv(ion.lib, sep = ",")
    }
    if (length(excludeLib) != 0) {
        for (i in 1:length(excludeLib)) {
            delFromLib <- grep(excludeLib[i], ion.lib[,1], fixed = TRUE, ignore.case=FALSE)
            ion.lib <- ion.lib[-delFromLib,]
        }
    }
    library_file <- ion.lib
    library_file$Name <- gsub(" ", "", library_file$Name)
    ## Extract ion ####
    extract_ion <- function(data, mz = 106, timeWindow = c(6,40), by = 0.01)
    {
        sample <- data
timeRange <- sample@scantime
timeRange <- round(timeRange)
    }
```
select <- timeRange[timeRange %in% seq(round(timeWindow[1]*60)-30, round(timeWindow[2]*60)+30, by)]

beggin <- which(timeRange == select[1])
beggin <- beggin[1]
finish <- which(timeRange == select[length(select)])
finish <- finish[length(finish)]

if(length(beggin) > 0 & length(finish) > 0){
  for (i in beggin:finish){
    feed <- getScan(sample, i)
    intensityOf <- data.frame(round(feed))
    intensityOf <- intensityOf[abs(intensityOf[,1] - mz) == 0,]
    if (nrow(intensityOf)>0){
      if (exists("listOf")){
        listOf <- c(listOf, max(intensityOf[,2]))
      } else {
        listOf <- max(intensityOf[,2])
      }
    } else {
      if (exists("listOf")){
        listOf <- c(listOf, 0)
      } else {
        listOf <- 0
      }
    }
  }

  rt <- sample@scantime[beggin:finish]/60
  listOf <- cbind(rt, intensity = as.numeric(listOf))
  return(data.frame(RT = listOf))
} else {
  return(data.frame(RT.rt = NA, RT.intensity = NA))
}

ion.lib[,2] <- as.numeric(as.character(ion.lib[,2]))

### Finish extract ion ####
color <- c("blue", "red", "orange")

for (c in 1:length(conditions)) {
  filenames <- dir(conditions[c])
  for (q in 1:length(filenames)) {
    file <- filenames[q]
    name.file <- gsub(".CDF", ", name.file, fixed = TRUE)
    data <- xcmsRaw(filename = paste(conditions[c], filenames[q], sep = ",")

    ### filtering noise ####
    toPlot <- data.frame(Intensity = data@scantime/60, RT = data@tic)
    plot(toPlot, type = "l")
    flatArea <- data@tic[duplicated(round(data@tic))]
    flatArea <- flatArea[flatArea != 0]
    flatArea <- min(flatArea)
    plotFlat <- data.frame(flatArea)
    plotFlat <- data.frame(flatArea[,1]/60, data@tic[flatArea[,1]])
    points(as.matrix(plotFlat[,1]), col = "red", pch = 19)
    segments(plotFlat[,1], plotFlat[,2], plotFlat[,1], max(data@tic)/2, col = "red", lwd = 2)
    text(plotFlat[,1], (max(data@tic)/2) + (max(data@tic)/12), "Noise subtraction point")
    scanValue <- getScan(data, flatArea[[1]])
    scanValue <- scanValue[scanValue[,2] != 0,]
  }
}

### Starts alignment ####
if (align == TRUE){
  if (internal.standard == ""){
    stop("No internal.standard defined. Please, define a internal standard or set align to FALSE.")
  }

  internalStd <- internal.standard
  internalStdRow <- which(ion.lib[,1] == internalStd)
  cpd1 <- ion.lib[internalStdRow,]
  ionFinalT <- extract_ion(data, as.numeric(cpd1[3]), c(as.numeric(cpd1[2]), as.numeric(cpd1[2])))
  }
ionFinalT <- ionFinalT[!is.na(ionFinalT[2]),]
names(ionFinalT)[2] <- as.character(cpd1[3])
if (filterNoise == TRUE){
  mzSub <- which(round(scanValue[,1]) == as.numeric(cpd1[3]))
  if (length(mzSub)>0){
    ionFinalT <- ionFinalT[ionFinalT[,2]>0,]
  }
}
ionFinalT <- ionFinalT[ionFinalT[,2] > 0,]
howMany <- nrow(ionFinalT)
if (howMany > 0){
  maxValue <- max(ionFinalT[2], na.rm = TRUE)
} else {
  maxValue <- 0
}
if (howMany > 0 & maxValue != 0){
  ## Smoothing graph ##
  ionFinal <- spline(ionFinalT[,1], ionFinalT[,2])
  ionFinal <- data.frame(ionFinal$x, ionFinal$y)
  names(ionFinal) <- c("RT.rt", as.character(cpd1[3]))
  ## End smoothing ###
  plot(as.matrix(ionFinal[1:2]), main = paste(name.file, as.character(cpd1[1,1]), sep = "-"))
  ## Find Peaks ###
  mainPeaks <- peaks(as.numeric(as.character(ionFinal[,1])), as.numeric(as.character(ionFinal[,2])), mode = "max")
  mainPeaks <- data.frame(Rt.rt = mainPeaks$x, mzName = mainPeaks$y)
  rowsIon <- which(round(as.numeric(as.character(ionFinalT[,1])),1) %in% round(as.numeric(as.character(mainPeaks[,1])),1))
  mainPeaks <- ionFinalT[rowsIon,]
  mainPeaks <- mainPeaks[!is.na(mainPeaks[1,]),]
  mainPeaks <- mainPeaks[mainPeaks[,2] != 0,]
  ## End find peaks ###
  if (nrow(mainPeaks) > 0){
    peakIonsRt <- mainPeaks[,1]
    for (j in 4:6){
      ionFinalT <- merge(ionFinalT, extract_ion(data, as.numeric(cpd1[j]), c(as.numeric(cpd1[2]), as.numeric(cpd1[2]))), by="RT.rt")
      names(ionFinalT)[j-1] <- as.character(cpd1[j])
      ionFinalT[is.na(ionFinalT)] <- 0
      if (filterNoise == TRUE){
        mzSub <- which(round(scanValue[,1]) == as.numeric(cpd1[j]))
        if (length(mzSub)>0){
          ionFinalT[j-1] <- ionFinalT[j-1] - scanValue[mzSub, 2]
          ionFinalT[ionFinalT < 0] <- 0
        }
      }
      lines(as.matrix(ionFinal), type = "p", pch = 1, lwd = 1, lty = 1, col = color[j-3])
    }
  }
  if (max(ionFinalT[,j-1], na.rm = TRUE) != 0){
    ## Smoothing graph ##
    mainPeaks <- peaks(as.numeric(as.character(ionFinal[,1])), as.numeric(as.character(ionFinal[,2])), mode = "max")
    mainPeaks <- data.frame(Rt.rt = mainPeaks$x, mzName = mainPeaks$y)
    rowsIon <- which(round(as.numeric(as.character(ionFinalT[,1])),1) %in% round(as.numeric(as.character(mainPeaks[,1])),1))
    mainPeaks <- ionFinalT[rowsIon,]
    mainPeaks <- mainPeaks[!is.na(mainPeaks[1,]),]
    mainPeaks <- mainPeaks[mainPeaks[,2] != 0,]
    peakIonsRt <- c(peakIonsRt, mainPeaks[,1])
  }
  missing <- apply(data.matrix(ionFinalT[3:5]), 1, function(x) sum(x == 0))
  ionFinalT <- ionFinalT[missing <= 1,]
  peakIonsRt <- peakIonsRt[peakIonsRt %in% ionFinalT[,1]]
  peakIonsRt <- count(data.frame(peakIonsRt))
peakIonsRt <- peakIonsRt[is.na(peakIonsRt[,1]),]
peakIonsRt <- peakIonsRt[peakIonsRt[,2]>=3,]
if (nrow(peakIonsRt)>0){
  for (j in 1:nrow(peakIonsRt)){
    peak1 <- ionFinalT[as.character(ionFinalT[,1]) == as.character(peakIonsRt[j,1]),]
    if (as.numeric(cpd1[7]) > (as.numeric(peak1[3])/as.numeric(peak1[2]))){
      inten2to1 <- (as.numeric(peak1[3])/as.numeric(peak1[2])) / as.numeric(cpd1[7])
    } else {
      inten2to1 <- as.numeric(cpd1[7]) / (as.numeric(peak1[3])/as.numeric(peak1[2]))
    }
    if (as.numeric(cpd1[8]) > (as.numeric(peak1[4])/as.numeric(peak1[2]))){
      inten3to1 <- (as.numeric(peak1[4])/as.numeric(peak1[2])) / as.numeric(cpd1[8])
    } else {
      inten3to1 <- as.numeric(cpd1[8]) / (as.numeric(peak1[4])/as.numeric(peak1[2]))
    }
    if (as.numeric(cpd1[9]) > (as.numeric(peak1[5])/as.numeric(peak1[2]))){
      inten4to1 <- (as.numeric(peak1[5])/as.numeric(peak1[2])) / as.numeric(cpd1[9])
    } else {
      inten4to1 <- as.numeric(cpd1[9]) / (as.numeric(peak1[5])/as.numeric(peak1[2]))
    }
    confirm <- c(inten2to1, inten3to1, inten4to1)
    confirm[is.na(confirm)] <- 0
    confirm <- count(confirm > matchFactor)
    confirm <- confirm[confirm[1] == TRUE,]
    if (nrow(confirm) == 0){
      peakIonsRt$Confirmed[j] <- 0
    } else {
      peakIonsRt$Confirmed[j] <- confirm$freq
    }
  }
  peakIonsRt <- peakIonsRt[peakIonsRt[,3] >= max(peakIonsRt[,3], na.rm = TRUE)-1,]
  if (nrow(peakIonsRt) > 0){
    peakIonsRt$score <- apply(peakIonsRt, 1, function(x) sum(x[2],x[3]))
  } else {
  }
  peakIonsRt <- peakIonsRt[peakIonsRt[,4] >= (max(peakIonsRt[,4], na.rm = TRUE)-1),]
  if (max(peakIonsRt[,4], na.rm = TRUE) >= 6){
    finalCheck <- ionFinalT[ionFinalT[,1] %in% peakIonsRt[,1],]
    ion1 <- finalCheck[finalCheck[,2] == max(finalCheck[,2], na.rm = TRUE),]
    ion2 <- finalCheck[finalCheck[,3] == max(finalCheck[,3], na.rm = TRUE),]
    ion3 <- finalCheck[finalCheck[,4] == max(finalCheck[,4], na.rm = TRUE),]
    ion4 <- finalCheck[finalCheck[,5] == max(finalCheck[,5], na.rm = TRUE),]
    extraPoints <- c(ion1[1,1], ion2[1,1], ion3[1,1], ion4[1,1])
    for (l in 1:length(extraPoints)){
      isThere <- which(peakIonsRt[,1] == extraPoints[l],]
      if (length(isThere) > 0){
        peakIonsRt[isThere, 4] <- peakIonsRt[isThere, 4] + 1
      }
    }
  } else {
  }
  peakIonsRt <- peakIonsRt[peakIonsRt[,4] >= max(peakIonsRt[,4], na.rm = TRUE),]
  if (nrow(peakIonsRt) > 1){
    peakIonsRt$rtdiff <- unlist(apply(peakIonsRt, 1, function(x) abs(x[1] - cpd1[2])))
    peakIonsRt <- peakIonsRt[peakIonsRt[,5] == min(peakIonsRt[,5]),]
    rtTime <- as.numeric(as.character(peakIonsRt[,1]))
    rtAlign <- as.numeric(as.character(cpd1[1,2])) - rtTime
    rtAlign <- rtAlign * 60
    data@scantime <- data@scantime + rtAlign
    print(paste("standard found at", rtTime, ", Chromatogram aligned!"))
  } else {
    rtTime <- as.numeric(as.character(peakIonsRt[,1]))
  }
  rtAlign <- as.numeric(as.character(cpd1[1,2])) - rtTime
  rtAlign <- rtAlign * 60
  data@scantime <- data@scantime + rtAlign
  print(paste("standard found at", rtTime, ", Chromatogram aligned!"))
} else {
  print("Chromatogram not aligned because no standard has been found")
}
```r
} else {
  print("Chromatogram not aligned because no standard has been found")
}
} else {
  print("Chromatogram not aligned because no standard has been found")
}
} else {
  print("Chromatogram not aligned because no standard has been found")
}
} else {
  print("Chromatogram not aligned because no standard has been found")
}
} else {
  print("Chromatogram not aligned because no standard has been found")
}
} else {
  print("Chromatogram not aligned because no standard has been found")
}
##### End of alignment ####
bar <- 0
toBar <- nrow(ion.lib)
for (i in 1:nrow(ion.lib)){
  Sys.sleep(0.0000000001)
  setTxtProgressBar(pb, i)
  cpd1 <- ion.lib[i,]
  ionFinalT <- extract_ion(data, as.numeric(cpd1[3]), c(as.numeric(cpd1[2]), as.numeric(cpd1[2])))
  ionFinalT <- ionFinalT[!is.na(ionFinalT[2]),]
  names(ionFinalT)[2] <- as.character(cpd1[3])
  if (filterNoise == TRUE){
    mzSub <- which(round(scanValue[,1]) == as.numeric(cpd1[3]))
    if (length(mzSub)>0){
      ionFinalT <- ionFinalT[ionFinalT[,2]>0,]
    }
  }
  ionFinalT <- ionFinalT[ionFinalT[,2] > 0,]
  howMany <- nrow(ionFinalT)
  if (howMany > 0){
    maxValue <- max(ionFinalT[2], na.rm = TRUE)
  } else {
    maxValue <- 0
  }
  if (howMany > 0 & maxValue != 0){
    #print("sec1")
    ## Smoothing graph ##
    ionFinal <- spline(ionFinalT[,1], ionFinalT[,2])
    ionFinal <- data.frame(ionFinal$x, ionFinal$y)
    names(ionFinal) <- c("RT.rt", as.character(cpd1[3]))
    ## End smoothing ###
    plot(as.matrix(ionFinal[1:2]), main = paste(name.file, as.character(cpd1[1,1]), sep = "-"))
    ## Find Peaks ###
    mainPeaks <- peaks(as.numeric(as.character(ionFinal[,1])), as.numeric(as.character(ionFinal[,2])), mode = "max")
    mainPeaks <- data.frame(Rt.rt = mainPeaks$X, mzName = mainPeaks$Y)
    rowsIon <- which(round(as.numeric(as.character(mainPeaks[,1])),1) %in% round(as.numeric(as.character(ionFinal[,1])),1))
    mainPeaks <- ionFinalT[rowsIon,]
    mainPeaks <- mainPeaks[!is.na(mainPeaks[,1]),]
    mainPeaks <- mainPeaks[mainPeaks[,2] != 0,]
    ## End find peaks ###
    if (nrow(mainPeaks) > 0){
      #print("sec2")
      peakIonsRt <- mainPeaks[,1]
      for (j in 4:6){
        ionFinalT <- merge(ionFinalT, extract_ion(data, as.numeric(cpd1[j]), c(as.numeric(cpd1[2]), as.numeric(cpd1[2])), by="RT.rt")
        names(ionFinalT)[j-1] <- as.character(cpd1[j])
        ionFinalT[is.na(ionFinalT)] <- 0
        if (filterNoise == TRUE){
          mzSub <- which(round(scanValue[,1]) == as.numeric(cpd1[j]))
        }
      }
    }
  }
  Sys.sleep(0.0000000001)
  setTxtProgressBar(pb, i)
  cpd1 <- ion.lib[i,]
  ionFinalT <- extract_ion(data, as.numeric(cpd1[3]), c(as.numeric(cpd1[2]), as.numeric(cpd1[2])))
  ionFinalT <- ionFinalT[!is.na(ionFinalT[2]),]
  names(ionFinalT)[2] <- as.character(cpd1[3])
  if (filterNoise == TRUE){
    mzSub <- which(round(scanValue[,1]) == as.numeric(cpd1[3]))
  }
```
if (length(mzSub)>0){
  ionFinalT[j-1] <- ionFinalT[j-1] - scanValue[mzSub, 2]
  ionFinalT[ionFinalT < 0] <- 0
}
lines(as.matrix(ionFinal), type = "p", pch = 1, lwd = 1, lty = 1, col = color[j-3])
if (max(ionFinalT[j-1], na.rm = TRUE) != 0){
  ## Smoothing graph ##
  ionFinal <- spline(ionFinalT[,1], ionFinalT[j-1])
  ionFinal <- data.frame(ionFinal$x, ionFinal$y)
  names(ionFinal) <- c("RT.rt", as.character(cpd1[3]))
  ## End smoothing ###
  mainPeaks <- peaks(as.numeric(as.character(ionFinal[,1])), as.numeric(as.character(ionFinal[,2])), mode = "max")
  mainPeaks <- data.frame(RT.rt = mainPeaks$x, mzName = mainPeaks$y)
  rowsIon <- which(round(as.numeric(as.character(ionFinalT[,1])),1) %in% round(as.numeric(as.character(mainPeaks[,1])),1))
  mainPeaks <- mainPeaks[!is.na(mainPeaks[1]),]
  mainPeaks <- mainPeaks[mainPeaks[,2] != 0,]
  peakIonsRt <- c(peakIonsRt, mainPeaks[,1])
  if (nrow(peakIonsRt)>0){
    #print("sec3")
    score <- 0
    for (j in 3:5){
      testCor <- cor(ionFinalT[,2], ionFinalT[,j])
      testCor[is.na(testCor)] <- 0
      if (testCor > correlation){
        score <- score + 1
      }
    }
    for (j in 1:nrow(peakIonsRt)){
      peak1 <- ionFinalT[as.character(ionFinalT[,1]) == as.character(peakIonsRt[j,1]),]
      if (as.numeric(cpd1[7]) > (as.numeric(peak1[3])/as.numeric(peak1[2]))){
        inten2to1 <- (as.numeric(peak1[3])/as.numeric(peak1[2])) / as.numeric(cpd1[7])
      } else {
        inten2to1 <- as.numeric(cpd1[7]) / (as.numeric(peak1[3])/as.numeric(peak1[2]))
      }
      if (as.numeric(cpd1[8]) > (as.numeric(peak1[4])/as.numeric(peak1[2]))){
        inten3to1 <- (as.numeric(peak1[4])/as.numeric(peak1[2])) / as.numeric(cpd1[8])
      } else {
        inten3to1 <- as.numeric(cpd1[8]) / (as.numeric(peak1[4])/as.numeric(peak1[2]))
      }
      if (as.numeric(cpd1[9]) > (as.numeric(peak1[5])/as.numeric(peak1[2]))){
        inten4to1 <- (as.numeric(peak1[5])/as.numeric(peak1[2])) / as.numeric(cpd1[9])
      } else {
        inten4to1 <- as.numeric(cpd1[9]) / (as.numeric(peak1[5])/as.numeric(peak1[2]))
      }
      confirm <- c(inten2to1, inten3to1, inten4to1, 0, inten3to2)
      confirm[is.na(confirm)] <- 0
      confirm <- count(confirm > matchFactor)
      confirm <- confirm[!is.na(confirm) == TRUE,]
      if (nrow(confirm) == 0){
        peakIonsRt$Confirmed[j] <- 0
      } else {
        peakIonsRt$Confirmed[j] <- confirm$freq
      }
    }
    peakIonsRt <- peakIonsRt[peakIonsRt[,3] >= max(peakIonsRt[,3], na.rm = TRUE)-1,]
    if (nrow(peakIonsRt) > 0){
      #print("sec4")
      ...
dupliRT <- duplicated(dupliRT)
if (length(dupliRT) > 0)
  for (d in 1:length(dupliRT))
    #print("sec8")
    #print(d)
    toDecide <- finalDf[finalDf[2] == dupliRT[d],]
    toDelete <- row.names(toDecide)
    #print(toDecide)
    #print(max(toDecide$Score, na.rm = TRUE))
    toDecide <- toDecide[toDecide$Score == max(toDecide$Score, na.rm = TRUE),]
    if (nrow(toDecide) > 1)
      toDecide <- toDecide[toDecide$diffTime == min(toDecide$diffTime, na.rm = TRUE),]
    } deleteComp <- which(toDelete != row.names(toDecide))
    #print(deleteComp)
    #print(toDelete[deleteComp])
  toDecide <- toDecide[toDecide$Score == max(toDecide$Score, na.rm = TRUE),]
  if (nrow(toDecide) > 1)
    toDecide <- toDecide[toDecide$diffTime == min(toDecide$diffTime, na.rm = TRUE),]
  deleteComp <- which(toDelete != row.names(toDecide))
    #print(deleteComp)
    #print(toDelete[deleteComp])
  finalDf <- finalDf[, which(row.names(finalDf) %in% row.names(toDelete[deleteComp]))]

finalDf <- finalDf[1:5]
#print(exists("finalDfT"))
#print(c)
#print(q)
#########################################################
if (!exists("finalDfT")){
  #print("sec9")
  #print(finalDf)
  finalDfT <- finalDf
delons(finalDfT)[2] <- name.file
  rm(finalDf)
} else {
  #print("sec10")
  #print(finalDf)
  finalDfT <- merge(finalDfT, finalDf, by = "Name", all = TRUE)
  delons(finalDfT)[delons(finalDfT)] <- name.file
  rm(finalDf)
  #print(finalDfT)
}

for (r in 1:nrow(replicates)) {
  rep.row <- replicates[r,]
  rep.row <- rep.row[!duplicated(rep.row)]
  rep.name <- rep(conditions[r], length(rep.row))
  if (r == 1) {
    rep.name.final <- c(rep.name)
    rep.name.final <- c("Replicates", rep.name.final)
  } else {
    rep.name.final <- c(rep.name.final, rep.name)
  }
  finalDfT[1] <- as.character(finalDfT[1,])
  finalDfT <- rbind(rep.name.final, finalDfT)
  if (save == TRUE) {
    sheet <- output
    store <- paste(main.folder, "/", sheet, ".csv", sep = "")
    write.csv(finalDfT, file = store, row.names = FALSE)
    print("Complete!")
    print(paste("The file ", output, ".csv", " was saved in the folder ", main.folder, sepe=""))
  } else {
    print("No file was saved because the argument save was set as FALSE")
  }
  return(finalDfT)
Appendix 3d.

R function extractIon.R for MacOS. This function is used to extract the intensity of specific fragments in samples analyzed by GC-MS. See section 3.4.2.

```
extractIon <- function(folder = tk_choose.dir(), compound = NA, ion.lib = tk_choose.files(multi=FALSE), timeWindow = NA, by = 0.01, save = TRUE, output = "fragment_intensity"){
  old.wd <- getwd()
  main.folder <- folder
  setwd(main.folder)
  files <- c(dir())
  info <- file.info(files)
  isDir <- info$isdir
  ## extract_ion ##
  if (is.data.frame(ion.lib) == FALSE) {
    ion.lib = read.csv(ion.lib, sep = ",")
  }
  if (is.na(compound)){
    compound <- select.list(c(as.character(ion.lib[,1]), "I want to define MZs"))
    if (compound != "I want to define MZs"){
      mzs <- which(ion.lib[1] == compound)
      rtWindow <- c(as.numeric(ion.lib[mzs,2])-0.3, as.numeric(ion.lib[mzs,2])+0.3)
      mzs <- c(ion.lib[mzs,3], ion.lib[mzs,4], ion.lib[mzs,5], ion.lib[mzs,6])
    } else {
      mzs <- select.list(c(1:1000), multiple=TRUE)
    }
    if (is.na(timeWindow)){
      timeWindow <- rtWindow
    }
  } else {
    mzs <- which(ion.lib[1] == compound)
    if (length(mzs) > 0){
      rtWindow <- c(as.numeric(ion.lib[mzs,2])-0.3, as.numeric(ion.lib[mzs,2])+0.3)
      mzs <- c(ion.lib[mzs,3], ion.lib[mzs,4], ion.lib[mzs,5], ion.lib[mzs,6])
    } else {
      stop("There is no compound with this name in the library you specified.")
    }
  }
  if (is.na(timeWindow)){
    timeWindow <- rtWindow
  }
  extract_ion <- function(data = choose.files(multi=FALSE), mzs = mzs, timeWindow = timeWindow, by = by)
  {
    sample <- xcmsRaw(data)
    timeRange <- sample@scantime
    timeRange <- round(timeRange)
    select <- timeRange[timeRange %in% seq(round(timeWindow[1]*60), round(timeWindow[2]*60), by)]
    beggin <- which(timeRange == select[1])
    beggin <- beggin+1
    finish <- which(timeRange == select[length(select)])
    finish <- finish[length(finish)]
    color = c("red", "blue", "orange", "black")
    pb <- txtProgressBar(min = 0, max = finish, style = 3, width = 50)
    for (i in beggin:finish)
  }
```
#print(i)
Sys.sleep(0.0000000001)
setTxtProgressBar(pb, i)
feed <- getScan(sample, i)
intensityOf <- data.frame(round(feed))
intensityOfFinal <- data.frame(mz = 0, intensity = 0)
for (j in 1:length(mzs)){
  intensityOf_each <- intensityOf[intensityOf[,1] == mzs[j],]
  if (nrow(intensityOf_each) > 0){
    intensityOf_each <- intensityOf_each[intensityOf_each[,2] == max(intensityOf_each[,2], na.rm = TRUE)]
    intensityOfFinal <- rbind(intensityOfFinal, intensityOf_each)
  } else {
    intensityOf_each <- data.frame(mz = mzs[j], intensity = 0)
    intensityOfFinal <- rbind(intensityOfFinal, intensityOf_each)
  }
}
names(intensityOfFinal)[2] <- round(sample@scantime[i]/60,5)
intensityOfFinal <- intensityOfFinal[-1,]
if (exists("listOf")){
  listOf <- merge(listOf, intensityOfFinal, by = "mz", all = TRUE)
} else {
  listOf <- intensityOfFinal
}
nameFile <- strsplit(data, split="\", fixed=TRUE)
nameFile <- unlist(nameFile)
maxY <- max(listOf[1], na.rm = TRUE)
plot(data.frame(RT = as.numeric(names(listOf[1])), Intensity = as.numeric(listOf[1,1])), col=color[1], main = nameFile[length(nameFile)], type = "b", pch = 1, lwd = 1, lty = 1, ylim=c(0, maxY))
for (i in 2:4){
  lines(data.frame(RT = as.numeric(names(listOf[1])), Intensity = as.numeric(listOf[i,1])), col=color[i], type = "b", pch = 1, lwd = 1, lty = 1)
}
legend("topleft", legend = c(listOf[1,1], listOf[2,1], listOf[3,1], listOf[4,1]), cex=0.6, bty="n", fill=color)
return(listOf)
}
## Finish extraction ##
conditions <- c(files[isDir == TRUE])
for (c in 1:length(conditions)) {
  filenames <- dir(conditions[c])
  for (q in 1:length(filenames)) {
    file <- filenames[q]
    name.file <- file
    extraction <- extract_ion(paste(conditions[c], filenames[q], sep = "/"), mzs, timeWindow, by)
    if (!exists("final.df")){
      final.df <- data.frame(t(c(name.file, extraction[1])))
    } else {
      final.df <- rbind(final.df, data.frame(t(c(name.file, extraction[1]))))
    }
    print(paste("File", name.file,"done!"))
    #forZoom <- readline()
    #if (forZoom == "z"){
    #  zoomplot(locator(2))
    #  readline()
    #}
    final.df2 <- data.frame(t(final.df[2]))
    names(final.df2) <- c(as.character(final.df[1,]))
    row.names(final.df2) <- paste("mz",as.character(mzs[1]))
    if (save == TRUE) {
      sheet <- output
    }
store <- paste(folder, "?", sheet, ",.csv", sep = "")
write.csv(final.df2, file = store, row.names = TRUE)
print("Complete!")
print(paste("The file ", output, ",.csv", " was saved in the folder ", folder, sep=""))
} else {
print("No file was saved because the argument save was set as FALSE")
}
return(final.df2)

########################################################################

Appendix 3e.

R function raw.peaks.R for MacOS. This function is used to extract the intensity of all fragments detected in samples analyzed by GC-MS. See section 3.4.2.
	num.rep <- length(grep(".CDF", c(list.files(conditions)[1],
    full.names = TRUE)), ignore.case = TRUE))
for (i in 2:length(conditions)) {
    num.rep <- c(num.rep, length(grep(".CDF", c(list.files(conditions)[i],
        full.names = TRUE)), ignore.case = TRUE))
    samples <- grep(".CDF", c(list.files(conditions)[i], full.names = TRUE)),
        ignore.case = TRUE, value = TRUE))
    replicates <- matrix(c(1:length(samples)), nrow = 1, ncol = length)
    for (i in 2:length(conditions)) {
        num.rep <- c(num.rep, length(grep(".CDF", c(list.files(conditions)[i],
            full.names = TRUE)), ignore.case = TRUE))
        samples <- c(samples, grep(".CDF", c(list.files(conditions)[i],
            full.names = TRUE)), ignore.case = TRUE, value = TRUE))
        up.num <- length(grep(".CDF", c(list.files(conditions)[i],
            full.names = TRUE)), ignore.case = TRUE, value = TRUE))
        if (i < 3) {
            low.num <- - (replicates[length(replicates)] + 1)
        } else {
            low.num <- max(replicates[i - 1, duplicated(replicates[i - 1,])]) + 1
        }
        replicates <- rbind(replicates, c((low.num):(low.num +
            up.num - 1))))
    }
    xset <- xcmsSet(samples)
    xset <- group(xset)
    if (correct.RT == TRUE) {
        xset <- retcor(xset, method = method)
        xset <- group(xset)
    }
Appendix 3f.

R function del.false.R for MacOS. This function is used to delete false positives in samples analyzed by GC-MS. See section 3.4.2.

del.false <- function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data"), multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character", true = 0.68, medium.tag = "none", true.medium = 0.68, save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."), output = "no_false") {
  if (is.data.frame(data) == TRUE) {
    data <- data
  } else {
    data = read.csv(data, colClasses = "character")
  }
  if (save == TRUE) {
    folder <- folder
  }
  if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {

  }
if (medium.tag == "none") {
    replicates <- as.character(data[1, ])
    rep <- 1
    reps <- factor(replicates[-1])
} else {
    medium <- which(medium.tag == data[1, ])
    if (length(medium) > 0) {
        medium.samples <- data[c(medium)]
        replicates.medium <- as.character(medium.samples[1, ])
        for (i in 1:nrow(medium.samples)) {
            if (sum(!is.na(medium.samples[i, ]))/ncol(medium.samples) <
                true.medium) {
                medium.samples[i, ] <- NA
            }
        }
        data[medium] <- medium.samples
    } else {
        stop("The medium.tag was not found in the row "Repl"
        ")
    }
    replicates <- as.character(data[1, ])
    rep <- 1
    reps <- factor(replicates[-1])
} } else {
    stop("There is no information about replicates in the data. The first row of the input data should contain the names of the conditions from where each sample belongs to. Try data(clean.fix.results) to see an example.")
} for (z in 1:length(levels(reps))) {
    if (as.character(levels(reps)[z]) == medium.tag) {
    } else {
        column <- which(as.character(levels(reps)[z]) ==
            as.character(data[1, ]))
        yMat <- data[column]
        for (i in 1:nrow(yMat)) {
            if (sum(!is.na(yMat[i, ]))/ncol(yMat) < true) {
                yMat[i, ] <- NA
            }
        }
        data[column] <- yMat
    }
} yMat <- data[-1, -1]
missing <- t(apply(yMat, 1, function(y) tapply(as.numeric(y),
    reps, function(x) sum(!is.na(x)))))
missing2 <- apply(t(apply(missing, 1, function(x) x == 0)),
    1, sum)
missing2 <- as.data.frame(missing2)
data <- merge(data, missing2, by = 0)
data <- subset(data, data$missing2 < length(levels(reps)))
data$missing2 <- NULL
data$Row.names <- NULL
if (rep == 1) {
    data <- rbind(c(replicates, NA), data)
} if (save == TRUE) {
    sheet <- output
    store <- paste(folder, ".", sheet, ".csv", sep = ")
    write.csv(data, file = store, row.names = FALSE)
} return(data)
Appendix 3g.

R function norm.internal.R for MacOS. This function is used to normalize metabolomics data by internal standard. See section 3.4.2.
	norm.internal <- function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"), internal.std = "ask", save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."), output = "norm_int")

{  
  if (is.data.frame(data) == TRUE) {
    data <- data
  } else {
    data = read.csv(data, colClasses = "character")
  }
  if (save == TRUE) {
    folder <- folder
  }
  if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
    replicates <- data[1, ]
    data <- data[-1, ]
    rep <- 1
  }
  standard <- "No"
  if (internal.std == "ask") {
    while (standard == "No") {
      internal <- select.list(data[, 1], title = "Select the internal standard:")
      internal <- grep(internal, data[, 1], value = FALSE, fixed = TRUE)
      print(data[internal, ])
      standard <- select.list(c("Yes", "No"), title = "Is it the right one?")
      if (standard == "Yes") {
        for (i in 2:ncol(data)) {
          if (is.na(data[internal, i])) {
            data[internal, i] <- "Not detected"
          } else {
            data[i] <- as.numeric(data[, i])/as.numeric(data[internal, i])
          }
        }
      } else {
        data[i] <- as.numeric(data[i])/as.numeric(data[internal, i])
      }
    }
  } else {
    internal <- grep(internal.std, data[, 1], value = FALSE)
    if (length(internal) == 0) {
      stop("There is no row containing the name you entered as compound or mass fragment to be used as internal standard.
    
    Please, try again but now entering the full name of the internal standard as it is written in your library.
  
  } else {
    internal <- grep(internal.std, data[, 1], value = FALSE)
    if (length(internal) == 0) {
      stop("There is no row containing the name you entered as compound or mass fragment to be used as internal standard.
    
    Please, try again but now entering the full name of the internal standard as it is written in your library.
  
  } else {
    if (length(internal) == 1) {
      for (i in 2:ncol(data)) {
        if (is.na(data[internal, i])) {
        
        
      } else {
        data[i] <- as.numeric(data[i])/as.numeric(data[internal, i])
      }
    }  
    }
Appendix 3h.

R function `norm.medium.R` for MacOS. This function is used to normalize metabolomics data by uncultured medium. See section 3.4.2.

```r
define function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"), medium = "popup", log.transform = TRUE, save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."), output = "norm_medium")
if (is.data.frame(data) == TRUE) {
data <- data
} else {
data <- read.csv(data, colClasses = "character"
} if (medium == "popup") {
medium <- readline("Tag for samples from the medium:
")
} if (save == TRUE) {
folder <- folder
} if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
replicates <- data[1, ]
data <- data[-1, ]
rep <- 1
reps <- factor(as.character(replicates[-1]))
} else {
stop("There is no information about replicates in the data. The first row of data should contain the information about replicates.")
} if (log.transform == TRUE) {
averages <- data.frame(t(apply(data[-1], 1, function(x) tapply(as.numeric(x), reps, function(x) mean(log(x), na.rm = TRUE)))))
med.col <- which(medium == names(averages))
averages <- averages[med.col]
med.col2 <- ncol(data) + 1
data[med.col2] <- averages
```
Appendix 3i.

R function norm.biomass.R for MacOS. This function is used to normalize metabolomics data by biomass content.

```r
norm.biomass <- function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data"), multi = FALSE, caption = "Select the CSV file containing the input data"),
  colClasses = "character"), biomass = read.csv(tk_choose.files(default = "Select the CSV file containing the biomasses."), multi = FALSE, caption = "Select the CSV file containing the biomasses."),
  colClasses = "character"), save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."),
  output = "norm_bio")
{
  if (!is.data.frame(data) || !is.data.frame(biomass)) {
    data <- rbind(replicates, data)
  }
  medium.col <- which(medium == data[1, ])
  data <- data[-(medium.col)]
  if (save == TRUE) {
    store <- paste(folder, "/", sheet, ".csv", sep = "")
    write.csv(data, file = store, row.names = FALSE)
  }
  return(data)
}
```

THE END
else {
    biomass = read.csv(biomass, colClasses = "character")
}
if (save == TRUE) {
    folder <- folder
}
if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
    replicates <- data[1, ]
    data <- data[-1, ]
    rep <- 1
} else {
bioData.sample <- biomass[biomass[1] == sample.name]
    if (length(bioData.sample) >= 1) {
        data[i] <- as.numeric(data[, i])/as.numeric(bioData.sample[2])
    } else {
        print(paste("Biomass was not found for", sample.name))
    }
    }
if (rep == 1) {
    data <- rbind(replicates, data)
}
if (save == TRUE) {
    sheet <- output
    store <- paste(folder, "/", sheet, ".csv", sep = ")
write.csv(data, file = store, row.names = FALSE)
return(data)
}

Appendix 3j.

R function htest.R for MacOS. This function is used to perform ANOVA or t-test on metabolomics data sets. See section 3.4.2.

htest <- function(data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"), signif.level = 0.05, log.transform = TRUE, save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved.", output = "htest")
{
    if (is.data.frame(data) == TRUE) {
        data <- data
    } else {
        data = read.csv(data, colClasses = "character")
    }
    if (save == TRUE) {
        folder <- folder
    }
    if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
        replicates <- as.character(data[1, ])
        data <- data[-1, ]
        rep <- 1
        reps <- factor(replicates[-1])
    }
else {
  stop("There is no information about replicates in the data. The first row of data should contain the information about replicates."")
}

if (length(levels(reps)) == 1) {
  stop("You defined only 1 condition in the first row of data. We need at least 2 conditions to run t-test or ANOVA."")
}

if (length(levels(reps)) > 2) {
  cat(paste("ANOVA is in progress..", "\n"))
  htest <- data[-1]
  missing <- t(apply(htest, 1, function(y) tapply(as.numeric(y),
                   reps, function(x) sum(is.na(x)))))
  missing2 <- apply(t(apply(missing, 1, function(x) x < 2)), 1, sum)
  htest2 <- htest[missing2 < (length(levels(reps)) - 1),]
  rest <- htest[missing2 %in% c((length(levels(reps)) - 1):(length(levels(reps)) - 1)),]
  if (nrow(rest) != 0) {
    rest$pvalues <- 0
  }
  if (log.transform == TRUE) {
    negative <- htest2
    negative[is.na(negative)] = 0
    negative <- data.frame(apply(negative, 1, function(x) sum(x < 0)))
    negative <- apply(negative, 2, function(x) sum(x > 0))
    if (negative > 0) {
      print("log.transform was set to TRUE. However, you have negative values in your input data and it is not possible to calculate the log of a negative value. Thus, the test will be performed with no log transformation.")
    }
  }
  else {
    htest2 <- data.matrix(htest2)
    htest2 <- log10(htest2)
  }
  lm.list <- apply(htest2, 1, function(x) lm(as.numeric(x) ~ reps))
  anovaLM.list <- lapply(lm.list, anova)
  pvalues <- data.frame(anovaLM.list)
  t <- 5
  htest2 <- data.frame(htest2)
  for (i in 1:nrow(htest2)) {
    htest2$pvalues[i] <- pvalues[1, t]
    t <- t + 5
  }
  htest3 <- rbind.data.frame(htest2, rest)
  data <- merge(data, htest3[ncol(hetest3)], by = 0)
  data <- subset(data, as.numeric(data$pvalues) < signif.level)
  data[1] <- NULL
  cat("Done.\n")
}
else {
  cat(paste("t-test is in progress..", "\n"))
  htest <- data[-1]
  missing <- t(apply(htest, 1, function(y) tapply(as.numeric(y),
                    reps, function(x) sum(is.na(x)))))
  missing2 <- apply(t(apply(missing, 1, function(x) x < 2)), 1, sum)
  htest2 <- htest[missing2 == 0,]
  rest <- htest[missing2 == 1,]
  if (nrow(rest) != 0) {
    rest$pvalues <- 0
  }
  if (log.transform == TRUE) {
    negative <- htest2
  }
  else {
    htest2 <- data[-1]
    missing <- t(apply(htest, 1, function(y) tapply(as.numeric(y),
                        reps, function(x) sum(is.na(x)))))
    missing2 <- apply(t(apply(missing, 1, function(x) x < 2)), 1, sum)
    htest2 <- htest[missing2 == 0,]
    rest <- htest[missing2 == 1,]
    if (nrow(rest) != 0) {
      rest$pvalues <- 0
    }
    if (log.transform == TRUE) {
      negative <- htest2
    }
  }
}
negative[is.na(negative)] = 0
negative <- data.frame(apply(negative, 1, function(x) sum(x < 0)))
negative <- apply(negative, 2, function(x) sum(x > 0))
if (negative > 0) {
print("log.transform was set to TRUE. However, you have negative values in your input data and it is not possible to calculate the log of a negative value. Thus, the test will be performed with no log transformation.")
} else {
htest2[is.na(htest2)] <- 0
htest2 <- data.matrix(htest2)
htest2 <- log10(htest2)
htest2[is.finite(htest2)] <- NA
}
list.p <- apply(htest2, 1, function(x) t.test(as.numeric(x) ~ reps))
list.p <- unlist(list.p)
t <- 3
htest2 <- data.frame(htest2)
for (i in 1:nrow(htest2)) {
htest2$pvalues[i] <- list.p[i]
t <- t + 11
}
htest3 <- rbind(htest2, rest)
data <- merge(data, htest3[ncol(htest3)], by = 0)
data <- subset(data, as.numeric(data$pvalues) < signif.level)
data[1] <- NULL
cat("Done.
")
if (rep == 1) {
data <- rbind(c(replicates, NA), data)
}
if (save == TRUE) {
sheet <- output
store <- paste(folder, "/", sheet, ".csv", sep = ")
write.csv(data, file = store, row.names = FALSE)
}
return(data)

冯
冯

Appendix 3k.

R function MetaBoxPCA.R for MacOS. This function is used to perform Principal Component Analysis on metabolomics data sets. See section 3.4.2.

MetaBoxPCA <- function(data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"), no.zeros = TRUE, plot.pcs = c(1,2)){
## Begin collecting arguments
if (is.data.frame(data) == TRUE) {
data <- data
} else {
data = read.csv(data, colClasses = "character")
}
if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
replicates <- as.character(data[1, ])
}
```r
data <- data[-1,]
rep <- 1
reps <- factor(replicates[-1])
}
names(data)[1] <- "Name"
## Finish collecting arguments
##
## Start PCA 2d
do.pca <- data
row.names(do.pca) <- do.pca$Name
do.pca$Name <- NULL
if (nlevels(reps) > 1){
  if (no.zeros == TRUE){
    missing <- t(apply(data.matrix(do.pca), 1, function(y) tapply(as.numeric(y), reps, function(x)
      sum(!is.na(x)))))
    do.pca <- subset(do.pca, apply(t(apply(missing, 1, function(x) x == 0)), 1, sum) == 0)
  }
}
do.pca <- do.pca[order(do.pca[1], decreasing=T),]
do.pca[is.na(do.pca)] <- 0
run.pca <- PCA(t(data.matrix(do.pca)), axes=plot.pcs)
## Finish PCA 2d

Appendix 4a.

R function papi.R for MacOS. See section 4.3.

papi <- function(data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"),
  save = TRUE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."),
  output = "papi_results"){
  ### Begin collecting arguments
  print("PAPI in progress...")
  if (is.data.frame(data) == TRUE) {
    omics.data.frame <- data
  } else {
    omics.data.frame = read.csv(data, colClasses = "character")
  }
  if (save == TRUE) {
    report_folder <- folder
  }
  if (omics.data.frame[1, 1] %in% c("Replicates", "Replicate", "replicates", 
    "replicate")) {
    replicates <- as.character(omics.data.frame[1,])
    omics.data.frame <- omics.data.frame[-1,]
    rep <- 1
    reps <- factor(replicates[-1])
  } else {
    stop("There is no information about replicates in the data. The first row of data should contain the information about replicates.")
    rep <- 0
  }
  ## Function to go to KEGG and collect pathways related to each compound
  getcomp <- function(x, y){
    comp <- melt(get.pathways.by.compounds(c(x, y)))
  }
  ## Function to go to KEGG and collect compounds playing part in each pathway
```

251
numbcomp <- function(x) {
  compounds <- melt(get.compounds.by.pathway(c(x)))
}

### Begin the PAPi algorithm
## Collect all pathways present in KEGG database
pathname <- data.frame(melt(as.list(KEGGPATHID2NAME)))
names(pathname)[c(1,2)] <- c("pathwayname", "idpathway")

## Choose column by column and apply PAPi
papi.frame <- numeric()
for(j in 2:ncol(omics.data.frame)){
  ## Prepare the list of compounds that will be searched in KEGG
  data.df <- omics.data.frame[-1,c(j)]
  data.df <- subset(data.df, !is.na(data.df[2]))
  complist <- as.character(data.df[,1])
  complist <- gsub("C", "cpd:C", complist, ignore.case = TRUE)

  ## Starts the progress bar
  pb <- txtProgressBar(min = 0, max = nrow(data.df), style = 3, width=50)

  ## Give to pathways the abundance of the related metabolite
  getpath.final <- numeric()
  for (i in 1:length(complist)){
    Sys.sleep(0.1)
    setTxtProgressBar(pb, i)
    getpath <- getcomp(complist[i], complist[i])
    if (nrow(getpath)>0){
      getpath$rate <- as.numeric(data.df[i,2])
      getpath.final <- getpath
    } else {
      getpath$rate <- as.numeric(data.df[i,2])
      getpath.final <- rbind(getpath, getpath.final)
    }
  }
  close(pb)

  ## Calculate the percentage of identified metabolites from each pathway
  path <- data.frame(getpath.final$value)
  getpath.final$value <- gsub("path:ko","", getpath.final$value)
  getpath.final$value <- gsub("path:map","", getpath.final$value)
  names(getpath.final)[1] <- "idpathway"
  result2 <- merge (getpath.final, pathname)

  ## Delete pathway "01100"
  metabolism <- grep("01100", result2$idpathway)
  result2 <- result2[-c(metabolism),]

  ## Delete pathway "01100"
  metabolism <- grep("01100", result2$idpathway)
  result2 <- result2[-c(metabolism),]

  result2 <- drop.levels(result2)
  result2$rate <- as.numeric(as.character(result2$rate))
  res.arr <- with(result2, tapply(rate, pathwayname, sum))
  res.df <- data.frame(pathwayname = names(res.arr), rate = res.arr)
  rownames(res.df) <- NULL
  pathwayfreq <- data.frame(table(result2$idpathway))
  names(pathwayfreq)[1] <- "idpathway"
  freqname <- merge (pathwayfreq, pathname)
  path <- as.character(freqname$idpathway)
  total <- nrow(freqname)

  ## Starts the second progress bar
  pb <- txtProgressBar(min = 0, max = total, style = 3, width=50)

  ## Starts the final calculations - Normalization
}
selecrow.final <- numeric()
for (i in 1:nrow(freqname)){
  Sys.sleep(0.1)
  setTxtProgressBar(pb, i)
  selecrow <- freqname[i,]
  map <- "path:ma"
  pathid <- c(map, path[i])
  allcpd <- numbcomp(paste(pathid, collapse = "p"))
  selecrow$percentage <- (selecrow$Freq/(nrow(allcpd)))
  if(length(selecrow.final) == 0){
    selecrow.final <- selecrow
  } else {
    selecrow.final <- rbind(selecrow.final, selecrow)
  }
}
close(pb)
selecrow.final <- selecrown.final[,-(1:2)]
final2.df <- merge(selecrow.final, res.df)
percent <- "percent"
names(final2.df)[3] <- names(data.df)[2]
final2.df[3] <- final2.df[3]/final2.df$percentage
final2.df$percentage <- NULL
confirmation <- paste("Sample", names(data.df), "done!", sep=" ")
print(confirmation)
if(length(papi.frame) == 0){
  papi.frame <- final2.df
} else {
  papi.frame <- merge(papi.frame, final2.df, all = TRUE)
}
## Finish PAPi algorithm
papi.frame[order(papi.frame[,2], decreasing = T),]
row.names(papi.frame) <- papi.frame[,1]
papi.frame[1] <- NULL
back <- papi.frame
if (rep == 1) {
  papi.frame <- rbind(c(replicates[-1]), papi.frame)
  row.names(papi.frame)[1] <- "Replicates"
}
## Save it in the specified folder
if (save == TRUE){
  sheet <- output
  store <- paste(folder, "/", sheet, ".csv", sep = " ")
  write.csv(papi.frame, file = store, row.names = TRUE)
  print(paste("The file ", output, ".csv", " was saved in the folder ", report_folder, sep=" "))
} else {
  print("No file was saved because the argument save was set as FALSE")
}
##
return(papi.frame)
}
################### THE END #########################

Appendix 4b.

R function papi.htest.R for MacOS. See section 4.3.

papi.htest <- function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data", multi = FALSE, caption = "Select the CSV file containing the input data"), colClasses = "character"), signif.level = 0.05, log.transform = TRUE,
save = FALSE, folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."), output = "htest")
{
if (is.data.frame(data) == TRUE) {
  data <- data
} else {
  data = read.csv(data, colClasses = "character")
}
if (save == TRUE) {
  folder <- folder
}
if (data[1, 1] %in% c("Replicates", "Replicate", "replicates","replicate")) {
  replicates <- as.character(data[1, ])
  data <- data[-1, ]
  rep <- 1
  reps <- factor(replicates[-1])
} else {
  stop("There is no information about replicates in the data. The first row of data should contain the information about replicates.")
}
if (length(levels(reps)) == 1) {
  stop("You defined only 1 condition in the first row of data. We need at least 2 conditions to run t-test or ANOVA.")
}
if (length(levels(reps)) > 2) {
  cat(paste("ANOVA is in progress. ", "un"))
  htest <- data[-1]
  missing <- t(apply(htest, 1, function(y) tapply(as.numeric(y), reps, function(x) sum(!is.na(x)))))
  missing2 <- apply(t(apply(missing, 1, function(x) x < 2)), 1, sum)
  htest2 <- htest[missing2 < (length(levels(reps)) - 1), ]
  rest <- htest[missing2 %in% c((length(levels(reps)) - 1):(length(levels(reps)) - 1)), ]
  if (nrow(rest) != 0) {
    rest$pvalues <- 0
  }
  if (log.transform == TRUE) {
    negative <- htest2
    negative[is.na(negative)] = 0
    negative <- data.frame(apply(negative, 1, function(x) sum(x < 0)))
    negative <- apply(negative, 2, function(x) sum(x > 0))
    if (negative > 0) {
      print("log.transform was set to TRUE. However, you have negative values in your input data and it is not possible to calculate the log of a negative value. Thus, the test will be performed with no log transformation.")
    } else {
      htest2 <- data.matrix(htest2)
      htest2 <- log10(htest2)
    }
  }
  lm.list <- apply(htest2, 1, function(x) lm(as.numeric(x) ~ reps))
  anovaLM.list <- lapply(lm.list, anova)
  pvalues <- data.frame(anovaLM.list)
  t <- 5
  htest2 <- data.frame(htest2)
  for (i in 1:nrow(htest2)) {
    htest2$pvalues[i] <- pvalues[1, t]
    t <- t + 5
  }
  htest3 <- rbind.data.frame(htest2, rest)
  data <- merge(data, htest3[ncol(htest3)], by = 0)
  data <- subset(data, as.numeric(data$pvalues) < signif.level)
  data[1] <- NULL
}
cat("Done.\n")
}

htest <- data[1]
missing <- t(apply(htest, 1, function(y) tapply(as.numeric(y),
    reps, function(x) sum(!is.na(x)))))
missing2 <- apply(t(apply(missing, 1, function(x) x < 2)), 1, sum)
htest2 <- htest[missing2 == 0,]
rest <- htest[missing2 == 1,]
if (nrow(rest) != 0) {
    rest$pvalues <- 0
}
if (log.transform == TRUE) {
    negative <- htest2
    negative[is.na(negative)] = 0
    negative <- data.frame(apply(negative, 1, function(x) sum(x < 0)))
    negative <- apply(negative, 2, function(x) sum(x > 0))
    if (negative > 0) {
        print("log.transform was set to TRUE. However, you have negative values in your input data and it is not possible to calculate the log of a negative value. Thus, the test will be performed with no log transformation."")
    } else {
        htest2[is.na(htest2)] <- 0
        htest2 <- data.matrix(htest2)
        htest2 <- log10(htest2)
        htest2[!is.finite(htest2)] <- NA
    }
}
list.p <- apply(htest2, 1, function(x) t.test(as.numeric(x) ~
    reps))
list.p <- unlist(list.p)
t <- 3
htest2 <- data.frame(htest2)
for (i in 1:nrow(htest2)) {
    htest2$pvalues[i] <- list.p[t]
    t <- t + 11
}
htest3 <- rbind(htest2, rest)
data <- merge(data, htest3[ncol(htest3)], by = 0)
data <- subset(data, as.numeric(data$pvalues) < signif.level)
data[1] <- NULL
cat("Done.\n")
}
if (rep == 1) {
    data <- rbind(c(replicates, NA), data)
}
if (save == TRUE) {
    sheet <- output
    store <- paste(folder, ",", sheet, ".csv", sep = "")
    write.csv(data, file = store, row.names = FALSE)
}
return(data)

###################### THE END ######################

Appendix 4c.

R function papi.line for MacOS. See section 4.3.

papi.line <- function (data = read.csv(tk_choose.files(default = "Select the CSV file containing the input data",

255
multi = FALSE, caption = "Select the CSV file containing the input data"),
colClasses = "character"), relative = TRUE,
folder = tk_choose.dir(caption = "Select the folder where the output file will be saved."),
output = "papi_line_graph", colour = "colors()", legend.position = c(0,0), cex.legend = 2,
cex.xlab = 0.8, cex.ylab = 0.8, position.ylab = c(-1,0), margins = c(30, 8, 2, 2),
yscale = "", dot.size = 2, graph.width=3000, graph.height=2000, graph.bg="transparent", graph.res = 300)
{
## Begin collecting arguments

report_folder <- folder
old.wd <- getwd()
setwd(report_folder)

if (is.data.frame(data) == TRUE) {
  data <- data
} else {
  data = read.csv(data, colClasses = "character")
}

if (data[1, 1] %in% c("Replicates", "Replicate", "replicates", "replicate")) {
  replicates <- as.character(data[1, ])
  data <- data[-1, ]
  rep <- 1
  reps <- factor(replicates[-1])
}

names(data)[1] <- "pathwayname"

## Finish collecting arguments

## Building graph

yMat <- data[-1]

## Build average column

avg <- t(apply(yMat, 1, function(y) tapply(as.numeric(y), reps, function(x) mean(x, na.rm = TRUE))))

if (nlevels(reps) == 1){
  avg <- as.data.frame(t(avg))
} else {
  avg <- as.data.frame(avg)
}

row.names(avg) <- data$pathwayname

## Start the scale invert process

if (nlevels(reps) > 1){
  if (relative == TRUE){
    for (j in 2:ncol(avg)){
      for (i in 1:nrow(avg)){
        if (ls.na(avg[i,1])){
          if (ls.na(avg[i,j])){
            if(avg[i,1] < avg[i,j]){ avg[i,j] <- (avg[i,j]/avg[i,1])
             } else { avg[i,j] <- (avg[i,j]/avg[i,1])
           }
         }
       }
     }
   }
   avg <- avg[order(avg[2], decreasing=TRUE),]
 }

## Reordering the data frame

## Building the plot

if (nlevels(reps) == 1){

names.row <- row.names(avg)
lines.new2 <- order(avg, decreasing=TRUE)
lines.new2 <- data.frame(avg[lines.new2])
lines.new2 <- cbind(names.row[1:lines.new2], lines.new2)
row.names(lines.new2) <- lines.new2$pathwayname
lines.new2$pathwayname <- NULL
avg <- lines.new2
if (colour[1] == "colors()"){
    colour <- colors()[30*c(1:nlevels(reps))]
} else {
    avg.g <- avg
    if (nlevels(reps) > 1){
        avg.g[1] <- 0
        avg.g[is.na(avg.g)] <- 0
    } else {
        avg <-. data.frame(avg)
        avg.g <- avg
    }
} if (nlevels(reps) > 1){
    if (nchar(yscale[1]) == 0){
        resumo <- summary(avg.g)
        yscale <- c(min(as.numeric(gsub(" \", "", (gsub("Min. \": "", resumo[,1]))))),
            max(as.numeric(gsub(" \", "", (gsub("Max. \": "", resumo[,6]))))))
    }
} if (nlevels(reps) == 1){
    if (nchar(yscale[1]) == 0){
        resumo <- summary(avg.g)
        yscale <- c(min(as.numeric(gsub(" \", "", (gsub("Min. \": "", resumo[,1]))))),
            max(as.numeric(gsub(" \", "", (gsub("Max. \": "", resumo[,6]))))))
    }
}

#################################################################
##### If more than two conditions #####
if (nlevels(reps) == 1){
    sheet <- paste(output, ".png", sep = "")
    store <- paste(report_folder, "/", sheet, sep="")
    png(store, width = graph.width, heigh = graph.width, bg = graph.bg, res = graph.res)
    par(mar=margins + 0.1, xpd = TRUE)
    plot(replicate(nrow(avg),0), type="l", lwd = 3, col=colour[1], axes=FALSE, ann=FALSE, lty=1, ylim=yscale)
    for (p in 1:nrow(avg)){
        points(p,as.matrix(avg[p,1]), col=colour[1], type="h")
    }
    axis(1, at=1:nrow(avg), lab=row.names(avg), las=2, cex.axis=cex.xlab, tck=-.01)
    axis(2, las=1, cex.axis=1, lwd.ticks = 1)
    text(position.ylab[1], position.ylab[2], labels = "Activity Score (AS)", cex = 1.5, srt = 90)
    legend(legend.position[1], legend.position[2], names(avg), cex=cex.legend, col=colour[1:ncol(avg)],
            lty = c(1,1,0,0), title="Conditions", lwd = 3, merge=FALSE)
    graphics.off()
}

#################################################################
##### If only two conditions ###
if (nlevels(reps) == 2){
    sheet <- paste(output, ".png", sep = "")
    store <- paste(report_folder, "/", sheet, sep="")
    png(store, width = graph.width, heigh = graph.width, bg = graph.bg, res = graph.res)
    par(mar=margins + 0.1, xpd = TRUE)
    plot(as.matrix(avg.g[1]), type="l", lwd = 3, col=colour[1], axes=FALSE, ann=FALSE, lty=1, ylim=yscale)
    for (p in 1:nrow(avg)){
        if (is.na(avg[p,1])){
            points(p,0, col=colour[2], type="o", pch = 19, cex = dot.size)
        } else {
            points(p,as.matrix(avg.g[p,1]), type="h", cex = 1.5, srt = 90)
        }
    }
    legend(legend.position[1], legend.position[2], names(avg), cex=cex.legend, col=colour[1:ncol(avg)],
            lty = c(1,1,0,0), title="Conditions", lwd = 3, merge=FALSE)
    graphics.off()
}
Appendix 5a.

Mean values ($n \geq 3$) and relative standard deviation (RSD) of relative abundances of metabolites detected in different samples of *L. lactis* cultivated on chemically defined media as described in chapter 5.

<table>
<thead>
<tr>
<th>Name</th>
<th>Glucose Intracellular</th>
<th>Glucose Intracellular</th>
<th>Glucose Extracellular</th>
<th>Glucose Extracellular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glucose Intracellular</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glucose Extracellular</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glucose Extracellular</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compound</td>
<td>Aerobic Average</td>
<td>Aerobic RSD (%)</td>
<td>Anaerobic Average</td>
<td>Anaerobic RSD (%)</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>2-Methyloctadecanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2-Phosphoenolpyruvic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Hydroxybenzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid</td>
<td>12.05</td>
<td>6.51</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3,5-Diido-L-tyrosine</td>
<td>65.97</td>
<td>23.38</td>
<td>0.07</td>
<td>38.77</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Alanine</td>
<td>168.21</td>
<td>19.54</td>
<td>60.70</td>
<td>13.92</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>64.21</td>
<td>15.80</td>
<td>6.11</td>
<td>17.28</td>
</tr>
<tr>
<td>Benzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C10:0 acid (Capric acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C12:0 acid (Lauric acid)</td>
<td>50.27</td>
<td>15.15</td>
<td>1.11</td>
<td>30.27</td>
</tr>
<tr>
<td>C14:0 acid (Myristic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C16:0 acid (Palmitic acid)</td>
<td>159.67</td>
<td>15.79</td>
<td>3.77</td>
<td>103.48</td>
</tr>
<tr>
<td>C16:1n-7 acid (Palmitoleic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C18:0 acid (Stearic acid)</td>
<td>43.48</td>
<td>48.28</td>
<td>0.83</td>
<td>46.64</td>
</tr>
<tr>
<td>C18:1n-9 acid (Oleic acid)</td>
<td>16.74</td>
<td>49.19</td>
<td>1.06</td>
<td>103.83</td>
</tr>
<tr>
<td>C18:2n-6,9 acid (Linoleic acid)</td>
<td>11.68</td>
<td>49.35</td>
<td>0.45</td>
<td>103.98</td>
</tr>
<tr>
<td>C18:3n-3,6,9 acid (Linolenic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C20:0 acid (Arachidic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C20:5n-3,6,9,12,15 acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Compound</td>
<td>1.36</td>
<td>17.97</td>
<td>0.28</td>
<td>11.70</td>
</tr>
<tr>
<td>-------------------</td>
<td>------</td>
<td>-------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td><strong>C21:0 acid</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Henelicosanoic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C22:0 acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>(Behenic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C6:0 acid</strong></td>
<td>15.21</td>
<td>6.26</td>
<td>0.21</td>
<td>33.35</td>
</tr>
<tr>
<td>(Capronic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>C8:0 acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>0.14</td>
<td>19.65</td>
</tr>
<tr>
<td>(Caprylic acid)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>cis-Aconitic acid</strong></td>
<td>0.86</td>
<td>33.96</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Citraconic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Citric acid</strong></td>
<td>45.29</td>
<td>18.63</td>
<td>8.61</td>
<td>9.63</td>
</tr>
<tr>
<td><strong>Cysteine</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Dodecane</strong></td>
<td>18.23</td>
<td>25.93</td>
<td>0.81</td>
<td>30.14</td>
</tr>
<tr>
<td><strong>Glutamic acid</strong></td>
<td>0.62</td>
<td>51.35</td>
<td>1.18</td>
<td>30.76</td>
</tr>
<tr>
<td><strong>Heptadecane</strong></td>
<td>NA</td>
<td>NA</td>
<td>1.08</td>
<td>10.89</td>
</tr>
<tr>
<td><strong>Histidine</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Isoleucine</strong></td>
<td>345.21</td>
<td>12.99</td>
<td>20.71</td>
<td>15.71</td>
</tr>
<tr>
<td><strong>Itaconic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>L-Norvaline</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>L-Ornithine</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Lactic acid</strong></td>
<td>206.31</td>
<td>57.03</td>
<td>51.75</td>
<td>25.43</td>
</tr>
<tr>
<td><strong>Leucine</strong></td>
<td>920.03</td>
<td>11.77</td>
<td>57.66</td>
<td>10.38</td>
</tr>
<tr>
<td><strong>Lysine</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Malonic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Methionine</strong></td>
<td>29.82</td>
<td>26.10</td>
<td>3.88</td>
<td>18.67</td>
</tr>
<tr>
<td><strong>N2-Acetyl-L-lysine</strong></td>
<td>215.67</td>
<td>11.88</td>
<td>0.73</td>
<td>30.00</td>
</tr>
<tr>
<td><strong>Nicotinic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>p-Toluic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Phenylalanine</strong></td>
<td>132.90</td>
<td>13.01</td>
<td>9.91</td>
<td>15.16</td>
</tr>
<tr>
<td><strong>Proline</strong></td>
<td>3491.37</td>
<td>15.01</td>
<td>182.10</td>
<td>13.42</td>
</tr>
<tr>
<td><strong>Pyroglutamic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Quinic acid</strong></td>
<td>0.94</td>
<td>35.25</td>
<td>6.14</td>
<td>21.30</td>
</tr>
<tr>
<td><strong>Serine</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td><strong>Succinic acid</strong></td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Name</td>
<td>2-Methyloctadecanoic acid</td>
<td>2-Phosphoenolpyruvic acid</td>
<td>3-Hydroxybenzoic acid</td>
<td>3-Hydroxybutyric acid</td>
</tr>
<tr>
<td>----------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td>----------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td></td>
<td>Galactose Intracellular Aerobic</td>
<td>Galactose Intracellular anaerobic</td>
<td>Galactose Extracellular Aerobic</td>
<td>Galactose Extracellular Anaerobic</td>
</tr>
<tr>
<td>Name</td>
<td>Averag e RSD (%)</td>
<td>Averag e RSD (%)</td>
<td>Averag e RSD (%)</td>
<td>Averag e RSD (%)</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>2-Methyloctadecanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2-Phosphoenolpyruvic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Hydroxybenzoic acid</td>
<td>1.43</td>
<td>23.30</td>
<td>0.49</td>
<td>28.54</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3,5-Diiodo-L-tyrosine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid</td>
<td>18.67</td>
<td>7.61</td>
<td>2.05</td>
<td>13.39</td>
</tr>
<tr>
<td>Alanine</td>
<td>388.18</td>
<td>11.02</td>
<td>200.81</td>
<td>33.01</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>12.62</td>
<td>21.31</td>
<td>21.13</td>
<td>38.68</td>
</tr>
<tr>
<td>Benzoic acid</td>
<td>6.64</td>
<td>19.17</td>
<td>1.55</td>
<td>43.36</td>
</tr>
<tr>
<td>C10:0 acid (Capric acid)</td>
<td>1.69</td>
<td>8.02</td>
<td>0.21</td>
<td>49.27</td>
</tr>
<tr>
<td>C12:0 acid (Lauric acid)</td>
<td>17.24</td>
<td>12.11</td>
<td>2.91</td>
<td>23.18</td>
</tr>
<tr>
<td>C14:0 acid (Myristic acid)</td>
<td>61.89</td>
<td>45.39</td>
<td>4.92</td>
<td>36.61</td>
</tr>
<tr>
<td>C16:0 acid (Palmitic acid)</td>
<td>98.80</td>
<td>45.99</td>
<td>7.63</td>
<td>40.16</td>
</tr>
<tr>
<td>C16:1n-7 acid (Palmitoleic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C18:0 acid (Stearic acid)</td>
<td>34.06</td>
<td>20.95</td>
<td>6.12</td>
<td>29.55</td>
</tr>
<tr>
<td>C18:1n-9 acid (Oleic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C18:2n-6,9 acid (Linoleic acid)</td>
<td>17.12</td>
<td>64.19</td>
<td>0.25</td>
<td>33.35</td>
</tr>
<tr>
<td>C18:3n-3,6,9 acid (Linolenic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C20:0 acid (Arachidic acid)</td>
<td>0.80</td>
<td>54.35</td>
<td>0.37</td>
<td>58.87</td>
</tr>
<tr>
<td>Compound</td>
<td>C20:5n-3,6,9,12,15 acid (EPA)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>C21:0 acid (Heneicosanoic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C22:0 acid (Behenic acid)</td>
<td>0.76</td>
<td>72.39</td>
<td>0.29</td>
<td>120.98</td>
</tr>
<tr>
<td>C6:0 acid (Capronic acid)</td>
<td>1.50</td>
<td>4.35</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C8:0 acid (Caprylic acid)</td>
<td>1.91</td>
<td>27.20</td>
<td>0.33</td>
<td>56.77</td>
</tr>
<tr>
<td>cis-Aconitic acid</td>
<td>2.69</td>
<td>11.50</td>
<td>1.51</td>
<td>98.75</td>
</tr>
<tr>
<td>Citraconic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Citric acid</td>
<td>122.52</td>
<td>28.31</td>
<td>58.04</td>
<td>70.75</td>
</tr>
<tr>
<td>Cysteine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Dodecane</td>
<td>6.20</td>
<td>12.23</td>
<td>0.75</td>
<td>13.27</td>
</tr>
<tr>
<td>Glutamic acid</td>
<td>4.01</td>
<td>28.11</td>
<td>8.37</td>
<td>15.08</td>
</tr>
<tr>
<td>Glycine</td>
<td>54.28</td>
<td>23.90</td>
<td>39.52</td>
<td>50.34</td>
</tr>
<tr>
<td>Heptadecane</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Histidine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>251.13</td>
<td>14.51</td>
<td>111.34</td>
<td>35.72</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>1.08</td>
<td>32.55</td>
<td>0.63</td>
<td>59.58</td>
</tr>
<tr>
<td>L-Norvaline</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>L-Ornithine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Lactic acid</td>
<td>428.82</td>
<td>10.85</td>
<td>150.42</td>
<td>21.72</td>
</tr>
<tr>
<td>Leucine</td>
<td>672.51</td>
<td>16.81</td>
<td>302.32</td>
<td>35.14</td>
</tr>
<tr>
<td>Lysine</td>
<td>45.44</td>
<td>18.76</td>
<td>56.06</td>
<td>80.41</td>
</tr>
<tr>
<td>Malonic acid</td>
<td>4.62</td>
<td>34.89</td>
<td>2.00</td>
<td>88.43</td>
</tr>
<tr>
<td>Methionine</td>
<td>54.32</td>
<td>19.39</td>
<td>24.52</td>
<td>28.07</td>
</tr>
<tr>
<td>N2-Acetyl-L-lysine</td>
<td>58.96</td>
<td>12.04</td>
<td>13.27</td>
<td>18.02</td>
</tr>
<tr>
<td>Nicotinic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>p-Toluic acid</td>
<td>6.75</td>
<td>8.33</td>
<td>1.46</td>
<td>34.91</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>207.62</td>
<td>17.76</td>
<td>99.64</td>
<td>31.95</td>
</tr>
<tr>
<td>Proline</td>
<td>1641.1</td>
<td>5</td>
<td>20.16</td>
<td>646.13</td>
</tr>
<tr>
<td>Pyroglutamic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Quinic acid</td>
<td>178.25</td>
<td>13.44</td>
<td>38.75</td>
<td>61.65</td>
</tr>
<tr>
<td>Serine</td>
<td>16.90</td>
<td>13.08</td>
<td>4.90</td>
<td>69.56</td>
</tr>
<tr>
<td>Succinic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Tartaric acid  
Threonine  
Tryptophan  
Tyrosine  
Valine

Appendix 5a. Continuation.

<table>
<thead>
<tr>
<th>Name</th>
<th>Lactose Intracellular Aerobic</th>
<th>Lactose Intracellular Anaerobic</th>
<th>Lactose Extracellular Aerobic</th>
<th>Lactose Extracellular Anaerobic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>RSD (%)</td>
<td>Average</td>
<td>RSD (%)</td>
</tr>
<tr>
<td>2-Methyloctadecanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2-Phosphoenolpyruvic acid</td>
<td>NA</td>
<td>NA</td>
<td>0.13</td>
<td>131.39</td>
</tr>
<tr>
<td>3-Hydroxybenzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Hydroxybutyric acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>3-Methyl-2-oxopentanoic acid</td>
<td>0.42</td>
<td>46.72</td>
<td>1.14</td>
<td>17.75</td>
</tr>
<tr>
<td>3,5-Diiodo-L-tyrosine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>4-Aminobenzoic acid</td>
<td>NA</td>
<td>NA</td>
<td>0.25</td>
<td>29.66</td>
</tr>
<tr>
<td>4-Methyl-2-oxopentanoic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Alanine</td>
<td>42.88</td>
<td>41.58</td>
<td>299.86</td>
<td>36.59</td>
</tr>
<tr>
<td>Aspartic acid</td>
<td>6.96</td>
<td>34.83</td>
<td>84.19</td>
<td>12.23</td>
</tr>
<tr>
<td>Benzoic acid</td>
<td>0.22</td>
<td>59.80</td>
<td>0.65</td>
<td>23.53</td>
</tr>
<tr>
<td>C10:0 acid (Capric acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C12:0 acid (Lauric acid)</td>
<td>0.48</td>
<td>30.00</td>
<td>33.59</td>
<td>14.11</td>
</tr>
<tr>
<td>C14:0 acid (Myristic acid)</td>
<td>2.17</td>
<td>28.89</td>
<td>44.85</td>
<td>27.74</td>
</tr>
<tr>
<td>C16:0 acid (Palmitic acid)</td>
<td>1.35</td>
<td>31.37</td>
<td>42.00</td>
<td>28.48</td>
</tr>
<tr>
<td>C16:1n-7 acid (Palmitoleic acid)</td>
<td>0.13</td>
<td>19.96</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C18:0 acid (Stearic acid)</td>
<td>0.32</td>
<td>39.02</td>
<td>49.80</td>
<td>20.37</td>
</tr>
<tr>
<td>C18:1n-9 acid (Oleic acid)</td>
<td>0.71</td>
<td>34.38</td>
<td>3.06</td>
<td>38.19</td>
</tr>
<tr>
<td>C18:2n-6,9 acid (Linoleic acid)</td>
<td>0.28</td>
<td>36.39</td>
<td>2.50</td>
<td>33.30</td>
</tr>
<tr>
<td>C18:3n-3,6,9 acid (Linolenic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Compound</td>
<td>NA</td>
<td>NA</td>
<td>1.72</td>
<td>47.99</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>----</td>
<td>----</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>C20:0 acid (Arachidic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C20:5n-3,6,9,12,15 acid (EPA)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C21:0 acid (Heneicosanoic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C22:0 acid (Behenic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C6:0 acid (Capronic acid)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>C8:0 acid (Caprylic acid)</td>
<td>0.09</td>
<td>27.09</td>
<td>2.33</td>
<td>18.87</td>
</tr>
<tr>
<td>cis-Aconitic acid</td>
<td>0.02</td>
<td>33.56</td>
<td>0.56</td>
<td>29.42</td>
</tr>
<tr>
<td>Citraconic acid</td>
<td>NA</td>
<td>NA</td>
<td>20.97</td>
<td>32.34</td>
</tr>
<tr>
<td>Citric acid</td>
<td>2.85</td>
<td>60.83</td>
<td>37.98</td>
<td>31.99</td>
</tr>
<tr>
<td>Cysteine</td>
<td>NA</td>
<td>NA</td>
<td>0.22</td>
<td>27.32</td>
</tr>
<tr>
<td>Dodecane</td>
<td>0.25</td>
<td>32.52</td>
<td>9.86</td>
<td>19.65</td>
</tr>
<tr>
<td>Glutamic acid</td>
<td>NA</td>
<td>NA</td>
<td>1.31</td>
<td>29.25</td>
</tr>
<tr>
<td>Glycine</td>
<td>16.02</td>
<td>52.46</td>
<td>80.57</td>
<td>28.94</td>
</tr>
<tr>
<td>Heptadecane</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Histidine</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Isoleucine</td>
<td>13.31</td>
<td>57.17</td>
<td>252.07</td>
<td>26.27</td>
</tr>
<tr>
<td>Itaconic acid</td>
<td>NA</td>
<td>NA</td>
<td>0.86</td>
<td>38.04</td>
</tr>
<tr>
<td>L-Norvaline</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>L-Ornithine</td>
<td>NA</td>
<td>NA</td>
<td>2.40</td>
<td>85.33</td>
</tr>
<tr>
<td>Lactic acid</td>
<td>12.62</td>
<td>60.50</td>
<td>417.39</td>
<td>30.76</td>
</tr>
<tr>
<td>Leucine</td>
<td>38.57</td>
<td>57.43</td>
<td>890.03</td>
<td>30.20</td>
</tr>
<tr>
<td>Lysine</td>
<td>1.72</td>
<td>77.92</td>
<td>6.11</td>
<td>49.91</td>
</tr>
<tr>
<td>Malonic acid</td>
<td>1.78</td>
<td>47.38</td>
<td>1.42</td>
<td>28.44</td>
</tr>
<tr>
<td>Methionine</td>
<td>1.65</td>
<td>55.15</td>
<td>16.56</td>
<td>73.96</td>
</tr>
<tr>
<td>N2-Acetyl-L-lysine</td>
<td>0.25</td>
<td>47.83</td>
<td>347.91</td>
<td>14.04</td>
</tr>
<tr>
<td>Nicotinic acid</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>p-Toluic acid</td>
<td>0.33</td>
<td>33.63</td>
<td>0.71</td>
<td>26.31</td>
</tr>
<tr>
<td>Phenylalanine</td>
<td>8.14</td>
<td>59.20</td>
<td>124.71</td>
<td>28.05</td>
</tr>
<tr>
<td>Proline</td>
<td>111.94</td>
<td>53.14</td>
<td>3221.9</td>
<td>5</td>
</tr>
<tr>
<td>Pyroglutamic acid</td>
<td>2.63</td>
<td>43.09</td>
<td>64.61</td>
<td>34.65</td>
</tr>
<tr>
<td>Quinic acid</td>
<td>1.45</td>
<td>26.48</td>
<td>0.83</td>
<td>31.64</td>
</tr>
<tr>
<td></td>
<td>accA_1</td>
<td>accB_1</td>
<td>accD_1</td>
<td>pycA_1</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-----------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>Gene code</strong></td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
<td>ATP + Pyruvate + HCO(3) + biotin &lt;=-&gt; ADP + phosphate + oxaloacetate</td>
</tr>
<tr>
<td><strong>Ecs</strong></td>
<td>6.4.1.2</td>
<td>6.4.1.2</td>
<td>6.4.1.2</td>
<td>6.4.1.1</td>
</tr>
</tbody>
</table>

**Appendix 6a.**

**Genome-scale metabolic network of Lactococcus lactis.**

<table>
<thead>
<tr>
<th>Gene code</th>
<th>Reactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>accA_1</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
</tr>
<tr>
<td>accB_1</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
</tr>
<tr>
<td>accD_1</td>
<td>ATP + Acetyl-CoA + HCO(3) + biotin =&gt; ADP + phosphate + Malonyl-CoA</td>
</tr>
<tr>
<td>pycA_1</td>
<td>ATP + Pyruvate + HCO(3) + biotin &lt;=-&gt; ADP + phosphate + oxaloacetate</td>
</tr>
<tr>
<td>pycA_2</td>
<td>ATP + Pyruvate + HCO(3) + biotin &lt;=-&gt; ADP + phosphate + oxaloacetate</td>
</tr>
<tr>
<td>pycA_3</td>
<td>ATP + Pyruvate + HCO(3) + biotin &lt;=-&gt; ADP + phosphate + oxaloacetate</td>
</tr>
<tr>
<td>pycA_4</td>
<td>ATP + Pyruvate + HCO(3) + biotin &lt;=-&gt; ADP + phosphate + oxaloacetate</td>
</tr>
<tr>
<td>carA_1</td>
<td>2 ATP + L-glutamine + HCO(3) + H(2)O =&gt; 2 ADP + phosphate + L-glutamate + carbamoyl phosphate + 2 H(+)</td>
</tr>
<tr>
<td>carB_1</td>
<td>2 ATP + L-glutamine + HCO(3) + H(2)O =&gt; 2 ADP + phosphate + L-glutamate + carbamoyl phosphate + 2 H(+)</td>
</tr>
<tr>
<td>asnB_1</td>
<td>ATP + L-aspartate + L-glutamine + H(2)O &lt;=&gt; AMP + diphosphate + L-asparagine + L-glutamate + H(+)</td>
</tr>
<tr>
<td>asnH_1</td>
<td>ATP + L-aspartate + L-glutamine + H(2)O &lt;=&gt; AMP + diphosphate + L-asparagine + L-glutamate + H(+)</td>
</tr>
<tr>
<td>purL_1</td>
<td>ATP + 5'-phosphoribosylformylglycinamidine + L-glutamine + H(2)O =&gt; ADP + phosphate + 5'-phosphoribosylformylglycinamidine + L-glutamate + H(+)</td>
</tr>
<tr>
<td>purQ_1</td>
<td>ATP + 5'-phosphoribosylformylglycinamidine + L-glutamine + H(2)O =&gt; ADP + phosphate + 5'-phosphoribosylformylglycinamidine + L-glutamate + H(+)</td>
</tr>
</tbody>
</table>
ATP + XMP + L-glutamine + H(2)O \rightarrow AMP + diphosphate + GMP + L-glutamate + 2 H(+)  
---

ATP + deamido-NAD(+) + L-glutamine + H(2)O \rightarrow AMP + diphosphate + L-argininosuccinate + H(+)  
---

ATP + L-citrulline + L-aspartate \rightarrow AMP + diphosphate + L-glutamine + H(2)O  
---

GTP + IMP + L-aspartate \rightarrow GDP + phosphate + adenylosuccinate + 2 H(+)  
---

ATP + formate + tetrahydrofolate \leftrightarrow ADP + phosphate + 10-formyltetrahydrofolate  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine + D-glutamate + L-lysine + H(2)O \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamate  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine + L-lysine \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamate + L-glutamate + H(2)O  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine + 1-(5-phosphoribosyl)-5-aminooimidazole  
---

ATP + 1-(5-phosphoribosyl)-5-aminooimidazole \rightarrow ADP + phosphate + 1-(5-phosphoribosyl)-4-(N-succinocarboxamide)-5-aminooimidazole  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine + D-glutamate \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamate + L-glutamate + H(2)O  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanine + L-lysine \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysine + H(2)O  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanyl-D-glutamate \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysine + H(2)O  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysine + D-alanyl-D-alanine \rightarrow ADP + phosphate + UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine  
---

ATP + UDP \rightarrow N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine + D-glutamate + NH(3) \rightarrow ADP + phosphate + L-glutamine  
---

ATP + 2-deoxy-D-ribose 1-phosphate \leftrightarrow 2-deoxy-D-ribose 5-phosphate  
---

ATP + 2-deoxy-D-ribose 5-phosphate \rightarrow D-ribose 1-phosphate  
---

ATP + beta-D-glucose 1-phosphate \leftrightarrow beta-D-glucose 6-phosphate  
---

ATP + 2-D-glucose 1-phosphate \rightarrow D-glucose 6-phosphate  
---
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Enzyme</th>
<th>EC Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-glucosamine-1-phosphate⇌D-glucosamine-6-phosphate</td>
<td>femD_2</td>
<td>5.4.2.10</td>
</tr>
<tr>
<td>3-phospho-D-glycerate⇌2-phospho-D-glycerate</td>
<td>pmg_1</td>
<td>5.4.2.1</td>
</tr>
<tr>
<td>3-phospho-D-glycerate⇌2-phospho-D-glycerate</td>
<td>yihF_1</td>
<td>5.4.2.1</td>
</tr>
<tr>
<td>3-phospho-D-glycerate⇌2-phospho-D-glycerate</td>
<td>yriJ_1</td>
<td>5.4.2.1</td>
</tr>
<tr>
<td>isopentenyl diphosphate⇌dimethylallyl diphosphate</td>
<td>yebB_1</td>
<td>5.3.3.2</td>
</tr>
<tr>
<td>D-glucose-6-phosphate⇌D-fructose-6-phosphate</td>
<td>pgiA_1</td>
<td>5.3.1.9</td>
</tr>
<tr>
<td>D-glucose-6-phosphate⇌beta-D-glucose-6-phosphate</td>
<td>pgiA_2</td>
<td>5.3.1.9</td>
</tr>
<tr>
<td>D-mannose-6-phosphate⇌D-fructose-6-phosphate</td>
<td>pmn_1</td>
<td>5.3.1.8</td>
</tr>
<tr>
<td>D-ribulose-5-phosphate⇌D-ribose-5-phosphate</td>
<td>rpiA_1</td>
<td>5.3.1.6</td>
</tr>
<tr>
<td>D-xylene ⇌ D-xylulose</td>
<td>xylA_1</td>
<td>5.3.1.5</td>
</tr>
<tr>
<td>D-galactose-6-phosphate⇌D-tagatose-6-phosphate</td>
<td>unk_10</td>
<td>5.3.1.26</td>
</tr>
<tr>
<td>N-(5-phospho-beta-D-ribose)-anthranilate⇌1-(2-carboxyphenylamino)-1-deoxy-D-ribulose-5-phosphate</td>
<td>trpF_1</td>
<td>5.3.1.24</td>
</tr>
<tr>
<td>D-glucuronate⇌D-fructuronate</td>
<td>uxaC_1</td>
<td>5.3.1.12</td>
</tr>
<tr>
<td>D-galacturonate⇌D-tagaturonate</td>
<td>uxaC_2</td>
<td>5.3.1.12</td>
</tr>
<tr>
<td>D-glucosamine-6-phosphate+H(2)O⇌D-fructose-6-phosphate+NH(3)</td>
<td>nagB_1</td>
<td>5.3.1.10</td>
</tr>
<tr>
<td>glycerone phosphate⇌D-glyceraldehyde-3-phosphate</td>
<td>tpiA_1</td>
<td>5.3.1.1</td>
</tr>
<tr>
<td>D-galactose⇌beta-D-galactose</td>
<td>galM_1</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>D-glucose⇌beta-D-glucose</td>
<td>xylM_1</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>D-xylene⇌beta-D-xylene</td>
<td>xylM_2</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>L-arabinose⇌beta-L-arabinose</td>
<td>xylM_3</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>lactose⇌beta-lactose</td>
<td>xylM_4</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>maltose⇌beta-maltose</td>
<td>xylM_5</td>
<td>5.1.3.3</td>
</tr>
<tr>
<td>UDP-galactose⇌UDP-glucose</td>
<td>galE_1</td>
<td>5.1.3.2</td>
</tr>
<tr>
<td>UDP-galactose⇌UDP-glucose</td>
<td>ycbD_1</td>
<td>5.1.3.2</td>
</tr>
<tr>
<td>dTDP-4-dehydro-6-deoxy-D-glucose⇌dTDP-4-dehydro-6-deoxy-L-mannose</td>
<td>cpsM_1</td>
<td>5.1.3.13</td>
</tr>
<tr>
<td>D-ribulose-5-phosphate⇌D-xylulose-5-phosphate</td>
<td>rpe_1</td>
<td>5.1.3.1</td>
</tr>
<tr>
<td>L-glutamate⇌D-glutamate</td>
<td>mruF_1</td>
<td>5.1.1.3</td>
</tr>
<tr>
<td>L-aspartate⇌D-aspartate</td>
<td>racD_1</td>
<td>5.1.1.13</td>
</tr>
<tr>
<td>L-alanine⇌D-alanine</td>
<td>dal_1</td>
<td>5.1.1.1</td>
</tr>
<tr>
<td>cystathionine+H(2)O⇌L-homocysteine+NH(3)+Pyruvate+H(+)</td>
<td>metC_1</td>
<td>4.4.1.8</td>
</tr>
</tbody>
</table>
unk_21  L-methionine + H(2)O => methanethiol + 2-oxobutanoate + NH(3)  4.4.1.11
unk_29  cystathionine + H(2)O => L-cysteine + NH(3) + 2-oxobutanoate  4.4.1.1
1-(5-phosphoribosyl)-4-(N-succino-carboxamido)- 5-aminomimidazole => fumarate + AICAR  4.3.2.2
purB_1  adenylosuccinate <= fumarate + AMP  4.3.2.2
purB_2  L-argininosuccinate <= fumarate + L-arginine  4.3.2.1
argH_1  L-threonine + H(2)O => 2-oxobutanoate + NH(3) + H(2)O  4.3.1.19
sdaA_1  L-serine <=> Pyruvate + NH(3)  4.3.1.17
sdaB_1  L-serine <=> Pyruvate + NH(3)  4.3.1.17
aroC_1  5-O-(1-carboxyvinyl)-3-phosphoshikimate => chorismate + phosphate  4.2.3.5
aroB_1  2-dehydro-3-deoxy-D-arabino-heptonate 7-phosphate => 3-dehydroquinate + phosphate  4.2.3.4
thrC_1  O-phospho-L-homoserine + H(2)O => L-threonine + phosphate  4.2.3.1
ilvD_1  2,3-dihydroxy-3-methylbutanoate => 3-methyl-2-oxobutanoate + H(2)O  4.2.1.19
ilvD_2  2,3-dihydroxy-3-methylpentanoate => 3-methyl-2-oxopentanoate + H(2)O  4.2.1.19
uxuA_1  D-mannonate <= 2-dehydro-3-deoxy-D-gluconate + H(2)O  4.2.1.8
uxaA_1  D-altronate => 2-dehydro-3-deoxy-D-gluconate + H(2)O  4.2.1.7
FabZ_1  (R)-3-hydroxybutanoyl-(acp) <=> But-2-enoyl-(acp) + H(2)O  4.2.1.58
dapA_1  L-aspartate 4-semialdehyde + Pyruvate => 2,3-dihydrodipicolinate + 2 H(2)O  4.2.1.52
pheA_1  prephenate + H(+) <=> phenylpyruvate + H(2)O + CO(2)  4.2.1.51
rmlB_1  dTDP-glucose => dTDP-4-dehydro-6-deoxy-D-glucose + H(2)O  4.2.1.46
leuC_1  2-isopropylmalate + H(2)O <= 3-isopropylmalate  4.2.1.33
leuD_1  2-isopropylmalate + H(2)O <= 2-isopropylmalate  4.2.1.33
citB_2  cis-aconitate + H(2)O => isocitrate  4.2.1.3
citB_3  citrate => cis-aconitate + H(2)O  4.2.1.3
unk_30  L-serine + L-homocysteine => cystathionine + H(2)O  4.2.1.22
trpA_1  L-serine + 1-(indol-3-yl)glycerol 3-phosphate <= L-tryptophan + D-glyceraldehyde 3-phosphate + H(2)O  4.2.1.20
trpB_1  L-serine + indole <= L-tryptophan + H(2)O  4.2.1.20
unk_6  fumarate + H(2)O <= (S)-malate  4.2.1.2
hisB_1  D-erythro-1-(imidazol-4-yl)glycerol 3-phosphate => 3-(imidazol-4-yl)-2-oxopropyl phosphate + H(2)O  4.2.1.19
enoA_1  2-phospho-D-glycerate <= phosphoenolpyruvate + H(2)O  4.2.1.11
enoB_1  2-phospho-D-glycerate <= phosphoenolpyruvate + H(2)O  4.2.1.11
aroD_1 3-dehydroquinate $\Leftrightarrow$ 3-dehydroshikimate + H(2)O 4.2.1.10
ribA_KEGG D-ribulose 5-phosphate $\Rightarrow$ 3,4-dihydroxy-2-butanone 4-phosphate + formate 4.1.99.12
citE_1 citrate $\Leftrightarrow$ acetate + oxaloacetate 4.1.3.6
pabB_1 chorismate + L-glutamine $\Rightarrow$ 4-amino-4-deoxychorismate + L-glutamate 4.1.3.38
pabB_2 4-amino-4-deoxychorismate $\Rightarrow$ 4-aminobenzoate + Pyruvate 4.1.3.38
menB_1 2-succinylbenzoyl-CoA $\Rightarrow$ 1,4-dihydroxy-2-naphthoate + CoA 4.1.3.36
trpE_1 chorismate + NH(3) $\Rightarrow$ anthranilate + Pyruvate + H(2)O 4.1.3.27
trpG_1 chorismate + L-glutamine $\Rightarrow$ anthranilate + Pyruvate + L-glutamate + H(+). 4.1.3.27
ptk_1 phosphate + D-xylulose 5-phosphate $\Leftrightarrow$ D-glyceraldehyde 3-phosphate + Acetyl-phosphate + H(2)O 4.1.2.9
unk_19 L-threonine $\Leftrightarrow$ glycine + acetaldehyde 4.1.2.5
unk_11 D-tagatose 1,6-bisphosphate $\Leftrightarrow$ glycerone phosphate + D-glyceraldehyde 3-phosphate 4.1.2.40
deoC_1 2-deoxy-D-ribose 5-phosphate $\Leftrightarrow$ D-glyceraldehyde 3-phosphate + acetaldehyde 4.1.2.4
folB_1 2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine $\Leftrightarrow$ 2-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine + glycolaldehyde 4.1.2.25
aroF_1 phosphoenolpyruvate + D-erythrose 4-phosphate + H(2)O $\Rightarrow$ 2-dehydro-3-deoxy-D-arabino-heptonate 7-phosphate + phosphate 4.1.2.15
aroH_1 phosphoenolpyruvate + D-erythrose 4-phosphate + H(2)O $\Rightarrow$ 2-dehydro-3-deoxy-D-arabino-heptonate 7-phosphate + phosphate 4.1.2.15
kdgA_1 2-dehydro-3-deoxy-D-gluconate 6-phosphate $\Rightarrow$ Pyruvate + D-glyceraldehyde 3-phosphate 4.1.2.14
fbA_1 D-fructose 1,6-bisphosphate $\Leftrightarrow$ glycerone phosphate + D-glyceraldehyde 3-phosphate 4.1.2.13
ipd_1 indol-3-pyruvate $\Rightarrow$ indol-3-acetaldehyde + CO(2) 4.1.1.74
aldB_1 2-acetolactate + H(+)$\Leftrightarrow$ 2-acetoin + CO(2) 4.1.1.5
trpC_1 1-(2-carboxyphenylamo)-1-deoxy-D-ribulose-5-phosphate $\Rightarrow$ 1-(indol-3-yl)glycerol 3-phosphate + CO(2) + H(2)O 4.1.1.48
pdc_1 phenylpyruvate $\Rightarrow$ phenylacetaldehyde + CO(2) 4.1.1.43
yeaH_1 ATP + (R)-5-diphosphomevalonate $\Rightarrow$ ADP + phosphate + isopentenyl diposphate + CO(2) 4.1.1.33
unk_16 oxaloacetate $\Rightarrow$ Pyruvate + CO(2) 4.1.1.3
pyrF_1 orotidine 5-phosphate $\Rightarrow$ UMP + CO(2) 4.1.1.23
purE_1 1-(5-phosphoribosyl)-5-aminimidazole + CO(2)$\Leftrightarrow$ 1-(5-phosphoribosyl)-5-aminimidazolecarboxylate 4.1.1.21
purK_1 1-(5-phosphoribosyl)-5-aminimidazole + CO(2)$\Leftrightarrow$ 1-(5-phosphoribosyl)-5-amin-4-imidazolecarboxylate 4.1.1.21
<table>
<thead>
<tr>
<th>Reaction</th>
<th>EC Number</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>lysA_1 meso-2,6-diaminohexanedioate + H(+) =&gt; L-lysine + CO(2)</td>
<td>4.1.1.20</td>
<td></td>
</tr>
<tr>
<td>gadB_1 L-glutamate + H(+) =&gt; 4-aminobutanoate + CO(2)</td>
<td>4.1.1.15</td>
<td></td>
</tr>
<tr>
<td>atpA ATP =&gt; ADP + phosphate</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpB 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpD 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpE 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpF 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpG 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>atpH 4 H(+)ext + ADP + phosphate &lt;=&gt; ATP + 3 H(+) + H(2)O</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>ATPp_1 ATP =&gt; ADP + phosphate</td>
<td>3.6.3.14</td>
<td></td>
</tr>
<tr>
<td>hisl_1 1-(5-phospho-D-ribosyl)-ATP + H(2)O =&gt; 1-(5-phospho-D-ribosyl)-AMP + diphosphate</td>
<td>3.6.1.31</td>
<td></td>
</tr>
<tr>
<td>dut_1 dUTP + H(2)O =&gt; dUMP + diphosphate</td>
<td>3.6.1.23</td>
<td></td>
</tr>
<tr>
<td>thiamine monophosphate + phosphate &lt;=&gt; thiamine diphosphate + H(2)O</td>
<td>3.6.1.15</td>
<td></td>
</tr>
<tr>
<td>DIPHOS diphosphate + H(2)O =&gt; 2 phosphate</td>
<td>3.6.1.1</td>
<td></td>
</tr>
<tr>
<td>tenA_1 thiamine + H(2)O =&gt; 4-amino-5-hydroxymethyl-2-methylpyrimidine + 4-methyl-5-(2-hydroxyethyl)-thiazole</td>
<td>3.5.99.2</td>
<td></td>
</tr>
<tr>
<td>cdd_1 cytidine + H(2)O &lt;=&gt; uridine + NH(3)</td>
<td>3.5.4.5</td>
<td></td>
</tr>
<tr>
<td>cdd_2 deoxycytidine + H(2)O =&gt; deoxyuridine + NH(3)</td>
<td>3.5.4.5</td>
<td></td>
</tr>
<tr>
<td>add_1 adenosine + H(2)O =&gt; inosine + NH(3)</td>
<td>3.5.4.4</td>
<td></td>
</tr>
<tr>
<td>add_2 deoxyadenosine + H(2)O =&gt; deoxyinosine + NH(3)</td>
<td>3.5.4.4</td>
<td></td>
</tr>
<tr>
<td>ribG_1 GTP + 3 H(2)O =&gt; formate + 2,5-diamino-6-hydroxy-4-(5-phosphoribosylamino)pyrimidine + H(2)O =&gt; 5-amino-6-(5-phosphoribosylamino)uracil + NH(3)</td>
<td>3.5.4.26</td>
<td></td>
</tr>
<tr>
<td>ribA_1 GTP + 3 H(2)O =&gt; formate + 2,5-diamino-6-hydroxy-4-(5-phosphoribosylamino)pyrimidine + diphosphate</td>
<td>3.5.4.25</td>
<td></td>
</tr>
<tr>
<td>hisl_2 1-(5-phospho-D-ribosyl)-AMP + H(2)O =&gt; N-(5'-phospho-D-ribosylformimino)-5-amino-1-(5''-phosphoribosyl)-4-imidazolcarboxamide</td>
<td>3.5.4.19</td>
<td></td>
</tr>
<tr>
<td>folE_1 GTP + 2 H(2)O =&gt; formate + 2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine triphosphate</td>
<td>3.5.4.16</td>
<td></td>
</tr>
<tr>
<td>dcdA_1 dCMP + H(2)O =&gt; dUMP + NH(3)</td>
<td>3.5.4.12</td>
<td></td>
</tr>
<tr>
<td>purH_2 5-formamido-1-(5-phospho-D-ribosyl)imidazole-4-carboxamide &lt;-&gt; IMP + H(2)O</td>
<td>3.5.4.10</td>
<td></td>
</tr>
<tr>
<td>arcA_1 L-arginine + H(2)O =&gt; L-citrulline + NH(3) + H(+)</td>
<td>3.5.3.6</td>
<td></td>
</tr>
<tr>
<td>pyrC_1 N-carbamoyl-L-aspartate + H(+) &lt;=&gt; (S)-dihydroorotate + H(2)O</td>
<td>3.5.2.3</td>
<td></td>
</tr>
<tr>
<td>unk_8 cysteamine + pantothenate &lt;=&gt; pantetheine + H(2)O</td>
<td>3.5.1.92</td>
<td></td>
</tr>
<tr>
<td>nagA_1 N-acetyl-D-glucosamine 6-phosphate + H(2)O &lt;=&gt; D-glucosamine 6-</td>
<td>3.5.1.25</td>
<td></td>
</tr>
</tbody>
</table>
phosphate + acetate

pepV_1
N-succinyl-LL-2,6-diaminoheptanedioate + H(2)O => succinate + LL-2,6-diaminoheptanedioate 3.5.1.18

argE_1
N2-acetyl-L-ornithine + H(2)O => acetate + L-ornithine 3.5.1.16

ansB_1
L-asparagine + H(2)O => L-aspartate + NH(3) + H(+) 3.5.1.1

bglA_1
6-phospho-beta-D-glucoside-(1,4)-D-glucose + H(2)O => D-glucose 6-phosphate + D-glucose 3.2.1.86

bglA_2
salicin 6-phosphate + H(2)O <=> beta-D-glucose 6-phosphate + 2-(hydroxymethyl)phenol 3.2.1.86

bglA_3
arbutin 6-phosphate + H(2)O <=> beta-D-glucose 6-phosphate + hydroquinone 3.2.1.86

xynB_1
1,4-beta-D-xylan + H(2)O <=> 2 D-xylose 3.2.1.37

xynB_2
xylolbios + H(2)O <=> 2 D-xylose 3.2.1.37

unk_3
sucrose 6-phosphate + H(2)O <=> D-fructose + D-glucose 6-phosphate 3.2.1.26

lacZ_1
lactose + H(2)O => D-galactose + D-glucose 3.2.1.23

agl_1
maltose + H(2)O => 2 D-glucose 3.2.1.20

agl_2
sucrose + H(2)O => D-glucose + D-fructose 3.2.1.20

dexA_1
alpha-dextrin => D-glucose 3.2.1.10

amyY_3
glycogen + H(2)O => maltose 3.2.1.1

amyY_4
glycogen + H(2)O => alpha-dextrin 3.2.1.1

amyY_5
bglcogen + H(2)O => maltose 3.2.1.1

amyY_6
glycogen + H(2)O => alpha-dextrin 3.2.1.1

amyL_3
glycogen + H(2)O => maltose 3.2.1.1

amyL_4
glycogen + H(2)O => alpha-dextrin 3.2.1.1

amyL_5
glycogen + H(2)O => maltose 3.2.1.1

amyL_6
glycogen + H(2)O => alpha-dextrin 3.2.1.1

acpD_1
ACP + H(2)O <=> pantetheine 4'-phosphate + apo-ACP 3.1.4.14

acpD_1_dup
ACP + H(2)O <=> pantetheine 4'-phosphate + apo-ACP 3.1.4.14

ydiD_1
ACP + H(2)O <=> pantetheine 4'-phosphate + apo-ACP 3.1.4.14

unk_65
adenosine 3',5'-bisphosphate + H(2)O => AMP + phosphate 3.1.3.7

nucA_1
AMP + H(2)O => adenosine + phosphate 3.1.3.5

nucA_10
dAMP + H(2)O => deoxyadenosine + phosphate 3.1.3.5

nucA_11
dGMP + H(2)O => deoxyguanosine + phosphate 3.1.3.5

nucA_2
IMP + H(2)O => inosine + phosphate 3.1.3.5

nucA_3
XMP + H(2)O => xanthosine + phosphate 3.1.3.5
nucA_4  \[
\text{GMP} + \text{H}_2\text{O} \rightarrow \text{guanosine} + \text{phosphate}
\]
3.1.3.5

nucA_5  \[
\text{CMP} + \text{H}_2\text{O} \rightarrow \text{cytidine} + \text{phosphate}
\]
3.1.3.5

nucA_6  \[
\text{UMP} + \text{H}_2\text{O} \rightarrow \text{uridine} + \text{phosphate}
\]
3.1.3.5

nucA_7  \[
\text{dTMP} + \text{H}_2\text{O} \rightarrow \text{thymidine} + \text{phosphate}
\]
3.1.3.5

nucA_8  \[
\text{dUMP} + \text{H}_2\text{O} \rightarrow \text{deoxyuridine} + \text{phosphate}
\]
3.1.3.5

nucA_9  \[
\text{dCMP} + \text{H}_2\text{O} \rightarrow \text{deoxycytidine} + \text{phosphate}
\]
3.1.3.5

serB_1  \[
\text{phosphoserine} + \text{H}_2\text{O} \rightarrow \text{L-serine} + \text{phosphate}
\]
3.1.3.3

pgpA_1  \[
\text{3-phosphatidylglycerol 1-phosphate} + \text{H}_2\text{O} \rightarrow \text{phosphatidylglycerol} + \text{phosphate}
\]
3.1.3.27

hisK_1  \[
\text{L-histidinol phosphate} + \text{H}_2\text{O} \rightarrow \text{L-histidinol} + \text{phosphate}
\]
3.1.3.15

fbp_1  \[
\text{D-fructose 1,6-bisphosphate} + \text{H}_2\text{O} \rightarrow \text{D-fructose 6-phosphate} + \text{phosphate}
\]
3.1.3.11

EF0365_1  \[
\text{dodecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{dodecanoate} + \text{ACP}
\]
3.1.2.14

EF0365_2  \[
\text{tetradecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{tetradecanoate} + \text{ACP}
\]
3.1.2.14

EF0365_3  \[
\text{hexadecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{hexadecenoate} + \text{ACP}
\]
3.1.2.14

EF0365_4  \[
\text{octadecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{octadecenoate} + \text{ACP}
\]
3.1.2.14

EF0365_5  \[
\text{octadecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{octadecanoate} + \text{ACP}
\]
3.1.2.14

EF0365_6  \[
\text{octadecanoyl-(ACP)} + \text{H}_2\text{O} \leftrightarrow \text{octadecanoate} + \text{ACP}
\]
3.1.2.14

acpS_1  \[
\text{CoA} + \text{apo-ACP} \rightarrow \text{adenosine 3',5'-bisphosphate} + \text{ACP}
\]
2.7.8.7

acpS_1_dup  \[
\text{CoA} \rightarrow \text{adenosine 3',5'-bisphosphate} + \text{ACP}
\]
2.7.8.7

pgsA_1  \[
\text{CDP-diacylglycerol} + \text{glycerol 3-phosphate} \leftrightarrow \text{CMP} + \text{3-phosphatidylglycerol 1-phosphate}
\]
2.7.8.5

pgsA_2  \[
\text{CDP-acylglycerol} + \text{glycerol 3-phosphate} \leftrightarrow \text{CMP} + \text{lyso phosphatidylglycerol 1-phosphate}
\]
2.7.8.5

mraY_1  \[
\text{UDP-N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine} + \text{undecaprenyl phosphate} \rightarrow \text{UMP} + \text{N-acetylmuramoyl-L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine-diphosphoundecaprenol}
\]
2.7.8.13

clsA_1  \[
2 \text{phosphatidylglycerol} \rightarrow \text{cardiolipin} + \text{glycerol}
\]
2.7.8.-

clsB_1  \[
2 \text{phosphatidylglycerol} \rightarrow \text{cardiolipin} + \text{glycerol}
\]
2.7.8.-

hasC_1  \[
\text{UTP} + \text{D-glucose 1-phosphate} \rightarrow \text{diphosphate} + \text{UDP-glucose}
\]
2.7.7.9

cdsA_1  \[
\text{CTP} + \text{diacylglycerol 3-phosphate} \rightarrow \text{diphosphate} + \text{CDP-diacylglycerol}
\]
2.7.7.41

cdsA_2  \[
\text{CTP} + \text{acylglycerol 3-phosphate} \rightarrow \text{diphosphate} + \text{CDP-acylglycerol}
\]
2.7.7.39

tagD1_1  \[
\text{CTP} + \text{glycerol 3-phosphate} + \text{H}(+) \rightarrow \text{diphosphate} + \text{CDP-glycerol}
\]
2.7.7.39

tagD2_1  \[
\text{CTP} + \text{glycerol 3-phosphate} + \text{H}(+) \rightarrow \text{diphosphate} + \text{CDP-glycerol}
\]
2.7.7.39

kdtB_1  \[
\text{ATP} + \text{pantetheine 4'-phosphate} + \text{H}(+) \rightarrow \text{diphosphate} + \text{diphospho-CoA}
\]
2.7.7.3

272
D-glucose 1-phosphate + ATP ⇄ ADP-D-glucose + diphosphate

D-glucose 1-phosphate + ATP ⇄ ADP-D-glucose + diphosphate

dTTP + D-glucose 1-phosphate ⇄ diphosphate + dTDP-glucose

UTP + N-acetyl-alpha-D-glucosamine 1-phosphate ⇄ diphosphate + UDP-N-acetyl-D-glucosamine

ATP + FMN ⇄ diphosphate + FAD

nicotinamide ribonucleotide + ATP ⇄ NAD(+) + diphosphate

nicotinate ribonucleotide + ATP ⇄ deamido-NAD(+) + diphosphate

UTP + alpha-D-galactose 1-phosphate ⇄ diphosphate + UDP-galactose

ATP + 2-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine ⇄ AMP + 2-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine
diphosphate

dATP + D-ribose 5-phosphate ⇄ dAMP + 5-phospho-alpha-D-ribose 1-diphosphate + 2 H(+)

ATP + D-ribose 5-phosphate ⇄ AMP + 5-phospho-alpha-D-ribose 1-diphosphate + 2 H(+)

ATP + dTMP ⇄ ADP + dTDP

ATP + dUTP ⇄ ADP + dUDP

ATP + dTMP ⇄ ADP + dTDP

ATP + dUTP ⇄ ADP + dUDP

ATP + GMP ⇄ ADP + GDP

ATP + dGMP ⇄ ADP + dGDP

ATP + 4-amino-2-methyl-5-phosphomethylpyrimidine ⇄ ADP + 4-amino-2-methyl-5-diphosphomethylpyrimidine

ATP + 4-amino-2-methyl-5-phosphomethylpyrimidine ⇄ ADP + 4-amino-2-methyl-5-diphosphomethylpyrimidine

GDP + ATP ⇄ GTP + ADP

ATP + dUDP ⇄ ADP + dUTP

ATP + AMP ⇄ 2 ADP

ATP + dAMP ⇄ ADP + dADP

ATP + (R)-5-phosphomevalonate ⇄ ADP + (R)-5-diphosphomevalonate

ATP + CMP ⇄ ADP + CDP

ATP + dCMP ⇄ ADP + dCDP

ATP + UMP ⇄ ADP + UDP

ATP + N-acetyl-L-glutamate ⇄ ADP + N-acetyl-5-glutamyl phosphate

ADP + 3-phospho-D-glyceroyl phosphate ⇄ ATP + 3-phospho-D-glycerate
arcC1_1  ADP + carbamoyl phosphate $\leftrightarrow$ ATP + NH$_3$ + CO$_2$  2.7.2.2
arcC2_1  ADP + carbamoyl phosphate $\leftrightarrow$ ATP + NH$_3$ + CO$_2$  2.7.2.2
arcC3_1  ADP + carbamoyl phosphate $\leftrightarrow$ ATP + NH$_3$ + CO$_2$  2.7.2.2
proB_1  ATP + L-glutamate $\rightarrow$ ADP + L-$\gamma$-glutamyl 5-phosphate  2.7.2.11
ackA1_1  ADP + Acetyl-phosphate $\rightarrow$ ATP + acetate  2.7.2.1
ackA2_1  ADP + Acetyl-phosphate $\rightarrow$ ATP + acetate  2.7.2.1
cb6p_1  cellobiose 6-phosphate $\leftrightarrow$ 6-phospho-beta-D-glucoside-(1,4)-D-glucose  2.7.1.85
dkB_2  ATP + deoxyadenosine $\rightarrow$ ADP + dAMP  2.7.1.76
udk_4  ATP + deoxycytidine $\rightarrow$ ADP + dCMP  2.7.1.74
aroK_1  ATP + shikimate $\rightarrow$ ADP + 3-phosphoshikimate  2.7.1.71
celB_1  CELBext + phosphoenolpyruvate $\rightarrow$ cellobiose 6-phosphate + Pyruvate  2.7.1.69
fruA_1  FRUtext + phosphoenolpyruvate $\rightarrow$ D-fructose 1-phosphate + Pyruvate  2.7.1.69
mtlF_1  GLUCext + phosphoenolpyruvate $\rightarrow$ D-glucose 6-phosphate + Pyruvate  2.7.1.69
mtlF_2  MNText + phosphoenolpyruvate $\rightarrow$ D-mannitol 1-phosphate + Pyruvate  2.7.1.69
ptbA_2  SALCext + phosphoenolpyruvate $\rightarrow$ salicin 6-phosphate + Pyruvate  2.7.1.69
ptbA_3  ARBtext + phosphoenolpyruvate $\rightarrow$ arbutin 6-phosphate + Pyruvate  2.7.1.69
ptcA_1  CELBext + phosphoenolpyruvate $\rightarrow$ cellobiose 6-phosphate + Pyruvate  2.7.1.69
ptcB_1  CELBext + phosphoenolpyruvate $\rightarrow$ cellobiose 6-phosphate + Pyruvate  2.7.1.69
ptcC_1  CELBext + phosphoenolpyruvate $\rightarrow$ cellobiose 6-phosphate + Pyruvate  2.7.1.69
ptnAB_1  GLUCext + phosphoenolpyruvate $\rightarrow$ D-glucose 6-phosphate + Pyruvate  2.7.1.69
ptnAB_2  MANnext + phosphoenolpyruvate $\rightarrow$ D-mannose 6-phosphate + Pyruvate  2.7.1.69
ptnAB_3  GLUMext + phosphoenolpyruvate $\rightarrow$ D-glucosamine 6-phosphate + Pyruvate  2.7.1.69
ptnAB_4  FRUtext + phosphoenolpyruvate $\rightarrow$ D-fructose 1-phosphate + Pyruvate  2.7.1.69
ptnAB_5  GLUMext + phosphoenolpyruvate $\rightarrow$ D-glucosamine 6-phosphate + Pyruvate  2.7.1.69
ptnC_1  MANnext + phosphoenolpyruvate $\rightarrow$ D-mannose 6-phosphate + Pyruvate  2.7.1.69
ptnC_2  MANnext + phosphoenolpyruvate $\rightarrow$ D-mannose 6-phosphate + Pyruvate  2.7.1.69
ptnC_3  GLUMext + phosphoenolpyruvate $\rightarrow$ D-glucosamine 6-phosphate + Pyruvate  2.7.1.69
ptnC_4  FRUtext + phosphoenolpyruvate $\rightarrow$ D-fructose 1-phosphate + Pyruvate  2.7.1.69
ptnC_5  GLUMext + phosphoenolpyruvate $\rightarrow$ D-glucosamine 6-phosphate + Pyruvate  2.7.1.69
ptnD_1  GLUCext + phosphoenolpyruvate $\rightarrow$ D-glucose 6-phosphate + Pyruvate  2.7.1.69
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Chemical Equations</th>
<th>Enzyme or Transporter</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptnD_2</td>
<td>MANNext + phosphoenolpyruvate =&gt; D-mannose 6-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>ptnD_3</td>
<td>GLUMext + phosphoenolpyruvate =&gt; D-glucosamine 6-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>ptnD_4</td>
<td>FRUText + phosphoenolpyruvate =&gt; D-fructose 1-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>ptnD_5</td>
<td>GLUMext + phosphoenolpyruvate =&gt; D-glucosamine 6-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>yleD_1</td>
<td>SUCRext + phosphoenolpyruvate =&gt; sucrose 6-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>yleD_2</td>
<td>TRHLext + phosphoenolpyruvate =&gt; trehalose 6-phosphate + Pyruvate</td>
<td>2.7.1.69</td>
</tr>
<tr>
<td>bacA_1</td>
<td>ADP + undecaprenyl phosphate + 2 H(+) &lt;=&gt; ATP + undecaprenol</td>
<td>2.7.1.66</td>
</tr>
<tr>
<td>galK_1</td>
<td>ATP + D-galactose =&gt; ADP + alpha-D-galactose 1-phosphate</td>
<td>2.7.1.6</td>
</tr>
<tr>
<td>pfk_2</td>
<td>D-fructose 1-phosphate + ATP =&gt; D-fructose 1,6-bisphosphate + ADP</td>
<td>2.7.1.56</td>
</tr>
<tr>
<td>thiM_1</td>
<td>ATP + 4-amino-5-hydroxymethyl-2-methylpyrimidine + ATP =&gt; ADP + 4-amino-2-methyl-5-phosphomethylpyrimidine</td>
<td>2.7.1.49</td>
</tr>
<tr>
<td>thiD_3</td>
<td>ATP + uridine =&gt; ADP + UMP</td>
<td>2.7.1.48</td>
</tr>
<tr>
<td>udk_1</td>
<td>ATP + cytidine =&gt; ADP + CMP</td>
<td>2.7.1.48</td>
</tr>
<tr>
<td>udk_2</td>
<td>ATP + 2-deoxy-3-deoxy-D-gluconate =&gt; ADP + 2-deoxy-3-deoxy-D-gluconate 6-phosphate + 2 H(+)</td>
<td>2.7.1.45</td>
</tr>
<tr>
<td>pyk_1</td>
<td>ADP + phosphoenolpyruvate =&gt; ATP + Pyruvate</td>
<td>2.7.1.40</td>
</tr>
<tr>
<td>scrK_1</td>
<td>ATP + D-fructose =&gt; ADP + D-fructose 6-phosphate</td>
<td>2.7.1.4</td>
</tr>
<tr>
<td>thrB_1</td>
<td>ATP + L-homoserine =&gt; ADP + O-phospho-L-homoserine</td>
<td>2.7.1.39</td>
</tr>
<tr>
<td>yeaG_1</td>
<td>ATP + (R)-mevalonate =&gt; ADP + (R)-5-phosphomevalonate</td>
<td>2.7.1.36</td>
</tr>
<tr>
<td>coaA_1</td>
<td>ATP + pantetheine =&gt; ADP + pantetheine 4'-phosphate</td>
<td>2.7.1.33</td>
</tr>
<tr>
<td>glpK_1</td>
<td>ATP + glycerol =&gt; ADP + glycerol 3-phosphate</td>
<td>2.7.1.30</td>
</tr>
<tr>
<td>ribC_1</td>
<td>ATP + riboflavin =&gt; ADP + FMN</td>
<td>2.7.1.26</td>
</tr>
<tr>
<td>unk_62</td>
<td>adenylyl sulfate + ATP =&gt; ADP + 3-phosphoadenylyl sulfate</td>
<td>2.7.1.25</td>
</tr>
<tr>
<td>yggA_1</td>
<td>ATP + dipospho-CoA =&gt; ADP + CoA + H(+)</td>
<td>2.7.1.24</td>
</tr>
<tr>
<td>udk_3</td>
<td>ATP + deoxyuridine =&gt; ADP + dUMP</td>
<td>2.7.1.21</td>
</tr>
<tr>
<td>yfG_1</td>
<td>ATP + thymidine =&gt; ADP + dTMP</td>
<td>2.7.1.21</td>
</tr>
<tr>
<td>glk_1</td>
<td>ATP + D-glucose =&gt; ADP + D-glucose 6-phosphate</td>
<td>2.7.1.2</td>
</tr>
<tr>
<td>glk_2</td>
<td>ATP + beta-D-glucose =&gt; beta-D-glucose 6-phosphate + ADP + 2 H(+)</td>
<td>2.7.1.2</td>
</tr>
<tr>
<td>xylB_1</td>
<td>ATP + D-xylulose =&gt; ADP + D-xylulose 5-phosphate + 2 H(+)</td>
<td>2.7.1.17</td>
</tr>
<tr>
<td>rbsK_2</td>
<td>ATP + D-ribose =&gt; ADP + D-ribose 5-phosphate</td>
<td>2.7.1.15</td>
</tr>
<tr>
<td>gntK_1</td>
<td>ATP + D-gluconate =&gt; ADP + 6-phospho-D-gluconate</td>
<td>2.7.1.12</td>
</tr>
</tbody>
</table>
dukA_1  ATP + deoxyguanosine => ADP + dGMP  2.7.1.113
dukA_2  ATP + deoxyadenosine => ADP + dAMP  2.7.1.113
dukB_1  ATP + deoxyguanosine => ADP + dGMP  2.7.1.113
pfk_1  ATP + D-fructose 6-phosphate => ADP + D-fructose 1,6-bisphosphate  2.7.1.11
dgkA_1  ATP + diacylglycerol => ADP + diacylglycerol 3-phosphate  2.7.1.107
lacC_1  D-tagatose 6-phosphate + ATP <=> D-tagatose 1,6-bisphosphate + ADP  2.7.1.1
hisC_1  3-(imidazol-4-yl)-2-oxopropyl phosphate + L-glutamate => L-histidinol phosphate + 2-oxoglutarate  2.6.1.9
pabA_1  chorismate + L-glutamine => 4-aminobenzoate + L-glutamate + Pyruvate  2.6.1.85
pabB_1  chorismate + L-glutamine => 4-aminobenzoate + L-glutamate + Pyruvate  2.6.1.85
pabA_2  chorismate + L-glutamine => 4-amino-4-deoxychorismate + L-glutamate  2.6.1.85
araT_11  L-phenylalanine + 4-methylthio 2-oxobutyrate <=> phenylpyruvate + L-methionine  2.6.1.60
araT_1  4-hydroxyphenylpyruvate + L-glutamate <=> L-tyrosine + 2-oxoglutarate  2.6.1.57
araT_7  phenylpyruvate + L-glutamate <=> L-phenylalanine + 2-oxoglutarate  2.6.1.57
serC_1  3-phosphohydroxyypyruvate + L-glutamate => phosphoserine + 2-oxoglutarate  2.6.1.52
unk_22  L-methionine + 2-oxoglutarate <=> 4-methylthio 2-oxobutyrate + L-glutamate  2.6.1.5
bcaT_1  L-valine + 2-oxoglutarate <=> 3-methyl-2-oxobutanoate + L-glutamate  2.6.1.42
bcaT_2  L-isoleucine + 2-oxoglutarate <=> 3-methyl-2-oxopentanoate + L-glutamate  2.6.1.42
bcaT_3  4-methyl-2-oxopentanoate + L-glutamate => L-leucine + 2-oxoglutarate  2.6.1.42
araT_10  L-phenylalanine + indol-3-pyruvate <=> phenylpyruvate + L-tryptophan  2.6.1.28
araT_16  L-tryptophan + phenylpyruvate <=> indol-3-pyruvate + L-phenylalanine  2.6.1.28
araT_13  L-tryptophan + 2-oxoglutarate <=> indol-3-pyruvate + L-glutamate  2.6.1.27
unk_18  Pyruvate + L-glutamate <=> L-alanine + 2-oxoglutarate  2.6.1.2
unk_31  N-succinyl-L-2-amino-6-oxoheptanedioate + L-glutamate <=> N-succinyl-LL-2,6-diaminoheptanedioate + 2-oxoglutarate  2.6.1.17
glmS_1  L-glutamine + D-fructose 6-phosphate => L-glutamate + D-glucosamine 6-phosphate  2.6.1.16
argD_1  N-acetyl-L-glutamate 5-semialdehyde + L-glutamate => N2-acetyl-L-ornithine + 2-oxoglutarate  2.6.1.11
aspB_1  oxaloacetate + L-glutamate <=> L-aspartate + 2-oxoglutarate  2.6.1.1
aspC_1  oxaloacetate + L-glutamate <=> L-aspartate + 2-oxoglutarate  2.6.1.1
ribH_1  2,6,7-dimethyl-8-(1-D-ribityl)lumazine => riboflavin + 4-(1-D-
ribozymino)-5-amino-2,6-dihydroxypryimidine

murA\textsubscript{1}
\begin{align*}
\text{phosphoenolpyruvate} & \rightarrow \text{UDP-N-acetyl-D-glucosamine} \\
& \rightarrow \text{phosphate} + \text{UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine}
\end{align*}
2.5.1.7

murA\textsubscript{2}
\begin{align*}
\text{phosphoenolpyruvate} & \rightarrow \text{UDP-N-acetyl-D-glucosamine} \\
& \rightarrow \text{phosphate} + \text{UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine}
\end{align*}
2.5.1.7

metK
\begin{align*}
\text{ATP} & \rightarrow \text{L-methionine} + \text{H}(2)\text{O} \\
& \rightarrow \text{phosphate} + \text{diphosphate} + \text{S-adenosyl-L-methionine}
\end{align*}
2.5.1.6

cysD\textsubscript{1}
\begin{align*}
\text{O-acetyl-L-homoserine} & \rightarrow \text{methanethiol} \\
& \rightarrow \text{L-methionine} + \text{acetate}
\end{align*}
2.5.1.49

cysD\textsubscript{2}
\begin{align*}
\text{O-acetyl-L-homoserine} & \rightarrow \text{H}(2)\text{S} \\
& \rightarrow \text{L-homocysteine} + \text{acetate}
\end{align*}
2.5.1.49

metB\textsubscript{1}
\begin{align*}
\text{O-succinyl-L-homoserine} & \rightarrow \text{cystathionine} + \text{succinate} + \text{H}(+) \\
& \rightarrow \text{phosphate} + \text{diphosphate}
\end{align*}
2.5.1.48

metB\textsubscript{2}
\begin{align*}
\text{O-succinyl-L-homoserine} & \rightarrow \text{cystathionine} + \text{succinate} + \text{H}(+) \\
& \rightarrow \text{phosphate} + \text{diphosphate}
\end{align*}
2.5.1.48

metB\textsubscript{1}
\begin{align*}
\text{NH}(3) & + \text{2-oxobutanoate} + \text{succinate} + \text{2 H}(+) \\
& \rightarrow \text{O-succinyl-L-homoserine} + \text{H}(2)\text{O}
\end{align*}
2.5.1.48

metB\textsubscript{2}
\begin{align*}
\text{O-acetyl-L-serine} & \rightarrow \text{H}(2)\text{S} \\
& \rightarrow \text{L-cysteine} + \text{acetate} + \text{H}(+)
\end{align*}
2.5.1.47

cysK\textsubscript{1}
\begin{align*}
\text{O-acetyl-L-serine} & \rightarrow \text{H}(2)\text{S} \\
& \rightarrow \text{L-cysteine} + \text{acetate} + \text{H}(+)
\end{align*}
2.5.1.47

cysM\textsubscript{1}
\begin{align*}
\text{O-acetyl-L-serine} & \rightarrow \text{H}(2)\text{S} \\
& \rightarrow \text{L-cysteine} + \text{acetate} + \text{H}(+)
\end{align*}
2.5.1.47

uppS\textsubscript{1}
\begin{align*}
8 \text{ isopentenyl diphosphate} & \rightarrow \text{trans-trans-farnesyl diphosphate} \\
& \rightarrow \text{undecaprenyl diphosphate} + \text{8 diphosphate}
\end{align*}
2.5.1.31

ispB\textsubscript{1}
\begin{align*}
4 \text{ isopentenyl diphosphate} & \rightarrow \text{trans-trans-farnesyl diphosphate} \\
& \rightarrow \text{trans-heptaprenyl diphosphate} + \text{4 diphosphate}
\end{align*}
2.5.1.30

thiE\textsubscript{1}
\begin{align*}
4 \text{ amino-2-methyl-5-diphosphomethylpyrimidine} & \rightarrow \text{4-methyl-5-(2-phosphoethyl)-thiazole} \\
& \rightarrow \text{diphosphate} + \text{thiamine monophosphate}
\end{align*}
2.5.1.15

aroA\textsubscript{1}
\begin{align*}
\text{phosphoenolpyruvate} & \rightarrow \text{3-phosphoshikimate} \\
& \rightarrow \text{phosphate} + 5\text{-O-(1-carboxyvinyl)-3-phosphoshikimate}
\end{align*}
2.5.1.19

folP\textsubscript{1}
\begin{align*}
2\text{-amino-4-hydroxy-6-hydroxymethyl-7,8-dihydropteridine diphosphate} & \rightarrow \text{diphosphate} + \text{dihydropterotate}
\end{align*}
2.5.1.15

ispA\textsubscript{1}
\begin{align*}
\text{dimethylallyl diphosphate} & \rightarrow \text{isopentenyl diphosphate} \\
& \rightarrow \text{geranyl diphosphate} + \text{diphosphate}
\end{align*}
2.5.1.10

ispA\textsubscript{2}
\begin{align*}
\text{geranyl diphosphate} & \rightarrow \text{isopentenyl diphosphate} \\
& \rightarrow \text{diphosphate} + \text{trans-trans-farnesyl diphosphate}
\end{align*}
2.5.1.10

ribH\textsubscript{KEGG}
\begin{align*}
3,4\text{-dihydroxy-2-butanoine} & \rightarrow \text{4-(1-D-ribitylamo)-5-amino-2,6-dihydroxypryimidine} \\
& \rightarrow \text{6,7-dimethyl-8-(1-D-ribityl)umazine} + \text{2 H}(2)\text{O} + \text{phosphate}
\end{align*}
2.5.1.7

ADPk\textsubscript{2}
\begin{align*}
\text{dADP} & \rightarrow \text{ATP} \\
& \rightarrow \text{dATP} + \text{ADP}
\end{align*}
2.4.7.6

hpt\textsubscript{1}
\begin{align*}
\text{IMP} & \rightarrow \text{diphosphate} \\
& \rightarrow \text{guanine} + \text{5-phospho-alpha-D-ribose 1-diphosphate}
\end{align*}
2.4.2.8

hprT\textsubscript{1}
\begin{align*}
\text{IMP} & \rightarrow \text{diphosphate} \\
& \rightarrow \text{hypoxanthine} + \text{5-phospho-alpha-D-ribose 1-diphosphate}
\end{align*}
2.4.2.8

hprT\textsubscript{2}
\begin{align*}
\text{GMP} & \rightarrow \text{diphosphate} \\
& \rightarrow \text{guanine} + \text{5-phospho-alpha-D-ribose 1-diphosphate}
\end{align*}
2.4.2.8

upp\textsubscript{1}
\begin{align*}
\text{uracil} & \rightarrow \text{5-phospho-alpha-D-ribose 1-diphosphate} \\
& \rightarrow \text{UMP} + \text{diphosphate}
\end{align*}
2.4.2.9

hpt\textsubscript{1}
\begin{align*}
\text{IMP} & \rightarrow \text{diphosphate} \\
& \rightarrow \text{hypoxanthine} + \text{5-phospho-alpha-D-ribose 1-diphosphate}
\end{align*}
2.4.2.8
murG_1
mapA_1
hisH_1
hisF_1
deoD_7
deoD_6
deoD_5
deoD_4
deoD_3
deoD_2
deoD_1
pyrE_1
xpt_1
udp_1
pdb_1
pdb_2
pdb_3
trpD_1
hisG_1
purF_1
unk_56
apt_1
udp_2
deoD_1
deoD_2
deoD_3
deoD_4
deoD_5
deoD_6
deoD_7
hisF_1
hisH_1
mapA_1
murG_1

GMP + diphosphate $\leftrightarrow$ guanine + 5-phospho-alpha-D-ribose 1-diphosphate

adene + 5-phospho-alpha-D-ribose 1-diphosphate $\leftrightarrow$ AMP + diphosphate

uridine + phosphate $\rightarrow$ uracil + D-ribose 1-phosphate

deoxyuridine + phosphate $\leftrightarrow$ uracil + 2-deoxy-D-ribose 1-phosphate

xanthine + 5-phospho-alpha-D-ribose 1-diphosphate $\leftrightarrow$ XMP + diphosphate

cytidine + phosphate $\leftrightarrow$ cytosine + D-ribose 1-phosphate

deoxyctydine + phosphate $\leftrightarrow$ cytosine + D-ribose 1-phosphate

thymine + 2-deoxy-D-ribose 1-phosphate $\leftrightarrow$ thymidine + phosphate

anthranilate + 5-phospho-alpha-D-ribose 1-diphosphate $\leftrightarrow$ N-(5-phospho-beta-D-ribosyl)-anthranilate + diphosphate

ATP + 5-phospho-alpha-D-ribose 1-diphosphate $\leftrightarrow$ 1-(5-phospho-D-riboyl)-ATP + diphosphate

L-glutamine + 5-phospho-alpha-D-ribose 1-diphosphate + H(2)O $\rightarrow$ 5-phospho-beta-D-riboylamine + diphosphate + L-glutamate

nicotinate + 5-phospho-alpha-D-ribose 1-diphosphate + H(+) $\rightarrow$ nicotinate ribonucleotide + diphosphate

orotate + 5-phospho-alpha-D-ribose 1-diphosphate $\leftrightarrow$ orotidine 5-phosphate + diphosphate

guanosine + phosphate $\leftrightarrow$ guanine + D-ribose 1-phosphate

inosine + phosphate $\leftrightarrow$ hypoxanthine + D-ribose 1-phosphate

deoxyguanosine + phosphate $\leftrightarrow$ guanine + 2-deoxy-D-ribose 1-phosphate

deoxyinosine + phosphate $\leftrightarrow$ hypoxanthine + 2-deoxy-D-ribose 1-phosphate

deoxyxanthosine + phosphate $\leftrightarrow$ xanthine + D-ribose 1-phosphate

deoxyadenosine + phosphate $\leftrightarrow$ adenine + 2-deoxy-D-ribose 1-phosphate

N-(5'-phospho-D-1'-ribulosylformimino)-5-amino-1-(5'-phosphoribosyl)-4-imidazolecarboxamide + L-glutamine $\leftrightarrow$ AICAR + D-erythro-1-(imidazol-4-y)glycerol 3-phosphate + L-glutamate

N-(5'-phospho-D-1'-ribulosylformimino)-5-amino-1-(5'-phosphoribosyl)-4-imidazolecarboxamide + L-glutamine $\leftrightarrow$ AICAR + D-erythro-1-(imidazol-4-y)glycerol 3-phosphate + L-glutamate

maltose + phosphate $\rightarrow$ D-glucose + beta-D-glucose 1-phosphate


2.4.2.8
2.4.2.7
2.4.2.3
2.4.2.3
2.4.2.22
2.4.2.2
2.4.2.2
2.4.2.2
2.4.2.18
2.4.2.17
2.4.2.14
2.4.2.11
2.4.2.2
2.4.2.1
2.4.2.1
2.4.2.1
2.4.2.1
2.4.2.8
2.4.2.1
2.4.2.1
2.4.2.1
2.4.1.227
glgA  ADP-D-glucose ⇌ ADP + glycogen + H(+)  2.4.1.21

glgB  glycogen ⇌ bglycogen  2.4.1.18

glgP_1  glycogen + phosphate ⇌ D-glucose 1-phosphate  2.4.1.1

glgP_2  bglycogen + phosphate ⇌ D-glucose 1-phosphate  2.4.1.1

leuA_1  Acetyl-CoA + 3-methyl-2-oxobutanoate + H(2)O ⇌ 2-isopropylmalate + CoA  2.3.3.13

hmcM_1  Acetyl-CoA + H(2)O + Acetoacetyl-CoA ⇌ (S)-3-hydroxy-3-methylglutaryl-CoA + CoA + H(+)  2.3.3.10

gltA_1  Acetyl-CoA + H(2)O + oxaloacetate ⇌ citrate + CoA + H(+)  2.3.3.1

fadA_1  2 Acetyl-CoA ⇌ CoA + Acetoacetyl-CoA  2.3.1.9

FabZ_2  (R)-3-hydroxyhexanoyl-(acp) ⇌ trans-hex-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_3  (R)-3-hydroxyoctanoyl-(acp) ⇌ trans-oct-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_4  (R)-3-hydroxydecanoyl-(acp) ⇌ trans-dec-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_5  (R)-3-hydroxydodecanoyl-(acp) ⇌ trans-dodec-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_6  (R)-3-hydroxytetradecanoyl-(acp) ⇌ trans-tetradec-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_7  (R)-3-hydroxyoctadecanoyl-(acp) ⇌ trans-octadec-2-enoyl-(acp) + H(2)O  2.3.1.85

FabZ_8  (R)-3-hydroxyoctadecanoyl-(acp) ⇌ trans-octadec-2-enoyl-(acp) + H(2)O  2.3.1.85

pta_1  Acetyl-CoA + phosphate ⇌ CoA + Acetyl-phosphate  2.3.1.8

maa_1  Acetyl-CoA + maltose ⇌ CoA + acetyl-maltose  2.3.1.79

pfl_1  CoA + Pyruvate ⇌ Acetyl-CoA + formate  2.3.1.54

metA_1  succinyl-CoA + L-homoserine ⇌ CoA + O-succinyl-L-homoserine  2.3.1.46

FabF_1  Acetyl-(acp) + Malonyl-(acp) ⇌ Acetoacetyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_2  Butyryl-(acp) + Malonyl-(acp) ⇌ 3-Oxohexanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_3  Hexanoyl-(acp) + Malonyl-(acp) ⇌ 3-Oxooctanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_4  Octanoyl-(acp) + Malonyl-(acp) ⇌ 3-oxodecanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_5  decanoyl-(acp) + Malonyl-(acp) ⇌ 3-Oxodecanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_6  dodecanoyl-(acp) + Malonyl-(acp) ⇌ 3-Oxotetradecanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_7  tetradecanoyl-(acp) + Malonyl-(acp) ⇌ 3-oxotetradecanoyl-(acp) + CO(2) + ACP  2.3.1.41

FabF_8  hexadecanoyl-(acp) + Malonyl-(acp) ⇌ 3-oxoheptadecanoyl-(acp) + CO(2) + ACP  2.3.1.41

fabD_1  Malonyl-CoA + ACP ⇌ Malonyl-(acp) + CoA  2.3.1.39
<table>
<thead>
<tr>
<th>Gene</th>
<th>Reaction</th>
<th>Enzyme Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>fabH_1</td>
<td>Acetyl-CoA + ACP $\rightarrow$ Acetyl-(acp) + CoA</td>
<td>2.3.1.38</td>
</tr>
<tr>
<td>argJ_2</td>
<td>N2-acetyl-L-ornithine + L-glutamate $\rightarrow$ L-ornithine + N-acetyl-L-glutamate</td>
<td>2.3.1.35</td>
</tr>
<tr>
<td>metX_1</td>
<td>L-homoserine + Acetyl-CoA $\Rightarrow$ O-acetyl-L-homoserine + CoA</td>
<td>2.3.1.31</td>
</tr>
<tr>
<td>cysE_1</td>
<td>Acetyl-CoA + L-serine $\leftrightarrow$ CoA + O-acetyl-L-serine</td>
<td>2.3.1.30</td>
</tr>
<tr>
<td>glmU_1</td>
<td>D-glucosamine 1-phosphate + Acetyl-CoA $\Rightarrow$ N-acetyl-alpha-D-glucosamine 1-phosphate + CoA</td>
<td>2.3.1.157</td>
</tr>
<tr>
<td>pdcC_1</td>
<td>CoA + S-acetyldihydrolipoamide $\Rightarrow$ Acetyl-CoA + dihydrolipoamide</td>
<td>2.3.1.12</td>
</tr>
<tr>
<td>ychH_1</td>
<td>succinyl-CoA + 2,3,4,5-tetrahydropyridine-2,6-dicarboxylate + H(2)O $\Rightarrow$ CoA + N-succinyl-L-2-amino-6-oxoheptanedioate</td>
<td>2.3.1.117</td>
</tr>
<tr>
<td>argJ_1</td>
<td>Acetyl-CoA + L-glutamate $\Rightarrow$ CoA + N-acetyl-L-glutamate + H(+)</td>
<td>2.3.1.1</td>
</tr>
<tr>
<td>menD_1</td>
<td>isochorismate + 2-oxoglutarate $\Rightarrow$ 2-succinyl-6-hydroxy-2,4-cyclohexadiene-1-carboxylate + Pyruvate + CO(2)</td>
<td>2.2.1.9</td>
</tr>
<tr>
<td>als_1</td>
<td>2 Pyruvate + H(+) $\Rightarrow$ 2-acetolactate + CO(2)</td>
<td>2.2.1.6</td>
</tr>
<tr>
<td>ilvB_1</td>
<td>2 Pyruvate + H(+) $\Rightarrow$ 2-acetolactate + CO(2)</td>
<td>2.2.1.6</td>
</tr>
<tr>
<td>ilvB_2</td>
<td>Pyruvate + 2-oxobutanoate + H(+) $\Rightarrow$ 2-aceto-2-hydroxybutanoate + CO(2)</td>
<td>2.2.1.6</td>
</tr>
<tr>
<td>ilvN_1</td>
<td>2 Pyruvate + H(+) $\Rightarrow$ 2-acetolactate + CO(2)</td>
<td>2.2.1.6</td>
</tr>
<tr>
<td>unk_2</td>
<td>sedoheptulose 7-phosphate + D-glyceraldehyde 3-phosphate $\leftrightarrow$ D-fructose 6-phosphate + D-erythrose 4-phosphate</td>
<td>2.2.1.2</td>
</tr>
<tr>
<td>tkt_1</td>
<td>D-ribose 5-phosphate + D-xylulose 5-phosphate $\leftrightarrow$ sedoheptulose 7-phosphate + D-glyceraldehyde 3-phosphate</td>
<td>2.2.1.1</td>
</tr>
<tr>
<td>tkt_2</td>
<td>D-erythrose 4-phosphate + D-xylulose 5-phosphate + thiamine diphosphate $\leftrightarrow$ D-fructose 6-phosphate + D-glyceraldehyde 3-phosphate</td>
<td>2.2.1.1</td>
</tr>
<tr>
<td>argF_1</td>
<td>carbamoyl phosphate + L-ornithine $\leftrightarrow$ phosphate + L-citrulline + H(+)</td>
<td>2.1.3.3</td>
</tr>
<tr>
<td>pyrB_1</td>
<td>carbamoyl phosphate + L-aspartate $\Rightarrow$ phosphate + N-carbamoyl-L-aspartate + H(+)</td>
<td>2.1.3.2</td>
</tr>
<tr>
<td>purH_1</td>
<td>10-formyltetrahydrofolate + AICAR $\Rightarrow$ tetrahydrofolate + 5-formamido-1-(5-phospho-D-ribosyl)imidazole-4-carboxamide</td>
<td>2.1.2.3</td>
</tr>
<tr>
<td>purN_1</td>
<td>10-formyltetrahydrofolate + 5'-phosphoribosylglycinamide $\Rightarrow$ tetrahydrofolate + 5'-phosphoribosylformylglycinamide</td>
<td>2.1.2.2</td>
</tr>
<tr>
<td>panB_1</td>
<td>3-methyl-2-oxobutanoate + 5,10-methylenetetrahydrofolate + H(2)O $\Rightarrow$ tetrahydrofolate + 2-dehydropanoate</td>
<td>2.1.2.11</td>
</tr>
<tr>
<td>glyA_1</td>
<td>5,10-methylenetetrahydrofolate + glycine + H(2)O $\Rightarrow$ tetrahydrofolate + L-serine</td>
<td>2.1.2.1</td>
</tr>
<tr>
<td>thyA_1</td>
<td>5,10-methylenetetrahydrofolate + dUMP $\Rightarrow$ dihydrofolate + dTMP</td>
<td>2.1.1.45</td>
</tr>
<tr>
<td>metE_1</td>
<td>5-methyltetrahydrofolate + L-homocysteine $\leftrightarrow$ tetrahydrofolate + L-methionine</td>
<td>2.1.1.13</td>
</tr>
</tbody>
</table>
| ubiE_1 | 2-demethylmenaquinone + S-adenosyl-L-methionine $\Rightarrow$ S-adenosyl-L-homocysteine + menaquinone + H(+) | 2.1.1-
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>trxB1_1</strong></td>
<td>NADPH + oxidized thioredoxin $\rightarrow$ NADP($^+$) + reduced thioredoxin</td>
</tr>
<tr>
<td><strong>trxB1_1</strong></td>
<td>NADPH + H($^+$) + oxidized thioredoxin $\rightarrow$ NADP($^+$) + reduced thioredoxin</td>
</tr>
<tr>
<td><strong>gshR_1</strong></td>
<td>NADPH + H($^+$) + oxidized glutathione $\rightarrow$ NADP($^+$) + 2 glutathione</td>
</tr>
<tr>
<td><strong>pdhD_1</strong></td>
<td>dihydrolipoamide + NAD($^+$) $\rightarrow$ lipoamide + NADH + H($^+$)</td>
</tr>
<tr>
<td><strong>noxA_1</strong></td>
<td>NADH + H($^+$) + O$_2$ $\rightarrow$ H$_2$O$_2$ + NAD($^+$)</td>
</tr>
<tr>
<td><strong>noxB_1</strong></td>
<td>NADH + H($^+$) + O$_2$ $\rightarrow$ H$_2$O$_2$ + NAD($^+$)</td>
</tr>
<tr>
<td><strong>guaC_1</strong></td>
<td>NADPH + GMP $\rightarrow$ NADP($^+$) + IMP + NH$_3$ + H($^+$)</td>
</tr>
<tr>
<td><strong>unk_48</strong></td>
<td>menaquinol $\leftrightarrow$ menaquinone + 2 H($^+$)</td>
</tr>
<tr>
<td><strong>folD_1</strong></td>
<td>5,10-methylenetetrahydrofolate + H$_2$O $\leftrightarrow$ 10-formyltetrahydrofolate + H($^+$)</td>
</tr>
<tr>
<td><strong>dfrA_1</strong></td>
<td>dihydrofolate + NADPH + H($^+$) $\rightarrow$ tetrahydrofolate + NADP($^+$)</td>
</tr>
<tr>
<td><strong>dfrA_2</strong></td>
<td>folate + NADPH + H($^+$) $\leftrightarrow$ dihydrofolate + NADP($^+$)</td>
</tr>
<tr>
<td><strong>folD_2</strong></td>
<td>5,10-methylenetetrahydrofolate + NADP($^+$) $\leftrightarrow$ 5,10-methylenetetrahydrofolate + NADPH</td>
</tr>
<tr>
<td><strong>metF_1</strong></td>
<td>5,10-methylenetetrahydrofolate + FADH(2) $\rightarrow$ 5-methyltetrahydrofolate + FAD</td>
</tr>
<tr>
<td><strong>proC_1</strong></td>
<td>1-pyrroline-5-carboxylate + NADPH + H($^+$) $\rightarrow$ L-proline + NADP($^+$)</td>
</tr>
<tr>
<td><strong>gltB_1</strong></td>
<td>L-glutamine + 2-oxoglutarate + NADPH + H($^+$) $\rightarrow$ 2 L-glutamate + NADP($^+$)</td>
</tr>
<tr>
<td><strong>gltD_1</strong></td>
<td>L-glutamine + 2-oxoglutarate + NADPH + H($^+$) $\rightarrow$ 2 L-glutamate + NADP($^+$)</td>
</tr>
<tr>
<td><strong>fodC_1</strong></td>
<td>succinate + FAD $\leftrightarrow$ fumarate + FADH(2)</td>
</tr>
<tr>
<td><strong>pydB_3</strong></td>
<td>(S)-dihydroorotate + fumarate $\leftrightarrow$ orotate + succinate</td>
</tr>
<tr>
<td><strong>pydB_2</strong></td>
<td>(S)-dihydroorotate + NAD($^+$) $\leftrightarrow$ orotate + NADH + H($^+$)</td>
</tr>
<tr>
<td><strong>FabI_1</strong></td>
<td>But-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Butyryl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_2</strong></td>
<td>trans-hex-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Hexanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_3</strong></td>
<td>trans-oct-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Octanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_4</strong></td>
<td>trans-dec-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Decanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_5</strong></td>
<td>trans-dodec-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Dodecanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_6</strong></td>
<td>trans-tetradec-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Tetradecanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_7</strong></td>
<td>trans-hexadec-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Hexadecanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>FabI_8</strong></td>
<td>trans-octadec-2-enoyl-(acp) + NADH + H($^+$) $\leftrightarrow$ Octadecanoyl-(acp) + NAD($^+$)</td>
</tr>
<tr>
<td><strong>dapB_1</strong></td>
<td>2,3-dihydrodipicolinate + NADPH + H($^+$) $\rightarrow$ 2,3,4,5-tetrahydroxyridine-2,6-dicarboxylate + NADP($^+$)</td>
</tr>
<tr>
<td>Name</td>
<td>Reaction</td>
</tr>
<tr>
<td>--------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>tyrA_1</td>
<td>prephenate + NAD(+) &lt;=&gt; 4-hydroxyphenylpyruvate + CO(2) + NADH + H(+)</td>
</tr>
<tr>
<td>unk_4</td>
<td>2-oxoglutarate + NAD(+) + CoA = succinyl-CoA + NADH + H(+) + CO(2)</td>
</tr>
<tr>
<td>pdhA_1</td>
<td>Pyruvate + lipoamide + H(+) = S-acetyldihydrolipoamide + CO(2)</td>
</tr>
<tr>
<td>pdhB_1</td>
<td>Pyruvate + lipoamide + H(+) = S-acetyldihydrolipoamide + CO(2)</td>
</tr>
<tr>
<td>pdhL_1</td>
<td>Pyruvate + phosphate + O(2) + H(2)O = Acetyl-phosphate + CO(2) + H(2)O(2)</td>
</tr>
<tr>
<td>proA_1</td>
<td>L-gamma-glutamyl 5-phosphate + NADPH + H(+) = L-glutamate 5-semialdehyde + phosphate + NADP(+)</td>
</tr>
<tr>
<td>argC_1</td>
<td>N-acetyl-L-glutamyl phosphate + NADPH + H(+) = N-acetyl-L-glutamate 5-semialdehyde + phosphate + NADP(+)</td>
</tr>
<tr>
<td>gapA_1</td>
<td>D-glyceraldehyde 3-phosphate + phosphate + NAD(+) = 3-phospho-D-glyceroyl phosphate + NADH + H(+)</td>
</tr>
<tr>
<td>gapB_1</td>
<td>D-glyceraldehyde 3-phosphate + phosphate + NAD(+) = 3-phospho-D-glyceroyl phosphate + NADH + H(+)</td>
</tr>
<tr>
<td>asd_1</td>
<td>4-phospho-L-aspartate + NADPH = L-aspartate 4-semialdehyde + phosphate + NADP(+)</td>
</tr>
<tr>
<td>adhE_1</td>
<td>Acetyl-CoA + NADH + H(+) = acetaldehyde + CoA + NAD(+)</td>
</tr>
<tr>
<td>nrdD_1</td>
<td>ATP + reduced thioredoxin = dATP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdD_2</td>
<td>GTP + reduced thioredoxin = dGTP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdD_3</td>
<td>CTP + reduced thioredoxin = dCTP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdD_4</td>
<td>UTP + reduced thioredoxin = dUTP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_1</td>
<td>ADP + reduced thioredoxin = dADP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_2</td>
<td>GDP + reduced thioredoxin = dGDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_3</td>
<td>CDP + reduced thioredoxin = dCDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_3_dup</td>
<td>CDP + reduced thioredoxin = dCDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_4</td>
<td>UDP + reduced thioredoxin = dUDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdE_4_dup</td>
<td>UDP + reduced thioredoxin = dUDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_1</td>
<td>ADP + reduced thioredoxin = dADP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_2</td>
<td>GDP + reduced thioredoxin = dGDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_3</td>
<td>CDP + reduced thioredoxin = dCDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_3_dup</td>
<td>CDP + reduced thioredoxin = dCDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_4</td>
<td>UDP + reduced thioredoxin = dUDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>nrdF_4_dup</td>
<td>UDP + reduced thioredoxin = dUDP + oxidized thioredoxin + H(2)O</td>
</tr>
<tr>
<td>sodA_1</td>
<td>2 superoxide + 2 H(+) = O(2) + H(2)O(2)</td>
</tr>
<tr>
<td>desA1_1</td>
<td>hexadecanoyl-(acp) + O(2) = hexadecenoyl-(acp) + 2 H(2)O</td>
</tr>
</tbody>
</table>
desA1_2  octadecanoyl-(acp) + O(2) ⇌ octadecenoyl-(acp) + 2 H(2)O  
1.14.19.2

gpo_1  2 glutathione + H(2)O(2) ⇒ oxidized glutathione + 2 H(2)O  
1.11.1.9

serA1_1  3-phospho-D-glycerate + NAD(+) ⇌ 3-phosphohydroxyphosphate + NADH + H(+)
1.1.1.95

gpdA1_1  glycerone phosphate + NADPH + H(+) ⇌ glycerol 3-phosphate + NADP(+) 
1.1.1.94

mvaA1_1  (S)-3-hydroxy-3-methylglutaryl-CoA + 2 NADPH + 2 H(+) ⇒ (R)-mevalonate + CoA + 2 NADP(+) 
1.1.1.88

ilvC1_1  2-acetolactate + NADPH + H(+) ⇒ 2,3-dihydroxy-3-methylbutanoate + NADP(+) 
1.1.1.86

ilvC2_2  2-aceto-2-hydroxybutanoate + NADPH + H(+) ⇒ 2,3-dihydroxy-3-methylpentanoate + NADP(+) 
1.1.1.86

leuB1_1  3-isopropylmalate + NAD(+) ⇒ 4-methyl-2-oxopentanoate + NADH + H(+) + CO(2)  
1.1.1.85

uxaB1_1  D-tagaturonate + NADH + H(+) ⇌ D-altronate + NAD(+)  
1.1.1.58

uxuB1_1  D-fructuronate + NADH + H(+) ⇌ D-mannonate + NAD(+) 
1.1.1.57

uxuB2_2  D-fructuronate + NADH + H(+) ⇌ D-mannonate + NAD(+)  
1.1.1.57

butA1_1  diacetyl + NADH + H(+) ⇒ 2-acetoin + NAD(+)  
1.1.1.5

zwf1_1  D-glucose 6-phosphate + NADP(+) ⇒ D-glucono-1,5-lactone 6-phosphate + NADPH + H(+)  
1.1.1.49

gnd1_1  6-phospho-D-glucuronate + NADP(+) ⇒ D-ribulose 5-phosphate + CO(2) + NADPH + H(+)  
1.1.1.44

gntZ1_1  6-phospho-D-glucuronate + NADP(+) ⇒ D-ribulose 5-phosphate + CO(2) + NADPH + H(+)  
1.1.1.44

icd1_1  isocitrate + NADP(+) ⇒ 2-oxoglutarate + CO(2) + NADPH + H(+)  
1.1.1.42

butB1_1  2-acetoin + NADH + H(+) ⇌ (R,R)-butane-2,3-diol + NAD(+)  
1.1.1.4

mae1_1  (S)-malate + NAD(+) ⇌ Pyruvate + CO(2) + NADH + H(+)  
1.1.1.38

maeP1_1  (S)-malate ⇒ (S)-lactate + CO(2)  
1.1.1.38

maeP1_1  MALAext + H(+)ext ⇌ (S)-malate + H(+)  
1.1.1.38

hom1_1  L-aspartate 4-semialdehyde + NADPH + H(+) ⇌ L-homoserine + NADP(+) 
1.1.1.3

thrA1_1  ATP + L-aspartate ⇒ ADP + 4-phospho-L-aspartate  
1.1.1.3

ldh1_1  Pyruvate + NADH + H(+) ⇒ (S)-lactate + NAD(+)  
1.1.1.27

ldhB1_1  Pyruvate + NADH + H(+) ⇒ (S)-lactate + NAD(+) 
1.1.1.27

ldhX1_1  Pyruvate + NADH + H(+) ⇒ (S)-lactate + NAD(+)  
1.1.1.27

aroE1_1  3-dehydroshikimate + NADPH + H(+) ⇒ shikimate + NADP(+)  
1.1.1.25

unk35_1  phenylpyruvate + NADH + H(+) ⇒ phenyllactate + NAD(+)  
1.1.1.237

hisD2_2  L-histidinol + NAD(+) ⇒ L-histidine + NADH + H(+) 
1.1.1.23
hisD_3: \[ \text{L-histidinal} + \text{NAD}^+ + \text{H}_2\text{O} \rightarrow \text{L-histidine} + \text{NADH} + \text{H}^+ \] 1.1.1.23

unk_33: 4-hydroxyphenylpyruvate + \text{NADH} \nleftrightarrow 4-hydroxyphenyllactate + \text{NAD}^+ 1.1.1.222

guaB_1: \[ \text{IMP} + \text{NAD}^+ + \text{H}_2\text{O} \rightarrow \text{XMP} + \text{NADH} + \text{H}^+ \] 1.1.1.205

ribG_2: \[ 5\text{-amino-6-(5-phosphoribosylamino)uracil} + \text{NADPH} + \text{H}^+ \rightarrow 5\text{-amino-6-(5-phosphoribitylamino)uracil} + \text{NADP}^+ \] 1.1.1.193

mtlD_1: \[ \text{D-mannitol 1-phosphate} + \text{NAD}^+ \nleftrightarrow \text{D-fructose 6-phosphate} + \text{NADH} + \text{H}^+ \] 1.1.1.17

panE_1: \[ (\text{R})\text{-pantoate} + \text{NADP}^+ \nleftrightarrow 2\text{-dehydropantoate} + \text{NADPH} + \text{H}^+ \] 1.1.1.169

murB_1: \[ \text{UDP-N-acetyl-3-O-(1-carboxyvinyl)-D-glucosamine} + \text{NADPH} + \text{H}^+ \rightarrow \text{UDP-N-acetylmuramate} + \text{NADP}^+ \] 1.1.1.158

rmlC_1: \[ \text{dTDP-4-dehydro-6-deoxy-L-mannose} + \text{NADPH} + \text{H}^+ \leftrightarrow \text{dTDP-rhamnose} + \text{NADP}^+ \] 1.1.1.133

FabG_1: \[ \text{Acetoacetyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxybutanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_2: \[ 3\text{-Oxohexanoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxyhexanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_3: \[ 3\text{-Oxooctanoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxyoctanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_4: \[ 3\text{-oxodecanoyl-(acp)} + \text{NADPH} \leftrightarrow (\text{R})\text{-3-hydroxydecanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_5: \[ 3\text{-Oxododecanoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxydodecanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_6: \[ 3\text{-oxotetradecanoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxytetradecanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_7: \[ 3\text{-oxohexadecanoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxyhexadecanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

FabG_8: \[ 3\text{-oxostearoyl-(acp)} + \text{NADPH} + \text{H}^+ \leftrightarrow (\text{R})\text{-3-hydroxyoctadecanoyl-(acp)} + \text{NADP}^+ \] 1.1.1.100

adhA_1: \[ \text{Acetaldehyde} + \text{NADH} + \text{H}^+ \rightarrow \text{Ethanol} + \text{NAD}^+ \] 1.1.1.1

thrA_2: \[ \text{ATP} + \text{L-aspartate} \rightarrow \text{ADP} + 4\text{-phospho-L-aspartate} \] 2.7.2.4

araT_12: \[ \text{L-phenylalanine} + 4\text{-hydroxyphenylpyruvate} \leftrightarrow \text{phenylpyruvate} + \text{L-tyrosine} \]

araT_14: \[ \text{L-tryptophan} + 4\text{-methyl-2-oxopentanoate} \leftrightarrow \text{indol-3-pyruvate} + \text{L-leucine} \]

araT_15: \[ \text{L-tryptophan} + 3\text{-methyl-2-oxopentanoate} \leftrightarrow \text{indol-3-pyruvate} + \text{L-isoleucine} \]

araT_17: \[ \text{L-tryptophan} + 4\text{-methylthio 2-oxobutyrate} \leftrightarrow 4\text{-hydroxyphenylpyruvate} + \text{L-methionine} \]

araT_18: \[ \text{L-tryptophan} + 4\text{-hydroxyphenylpyruvate} \leftrightarrow \text{indol-3-pyruvate} + \text{L-tyrosine} \]

araT_2: \[ \text{L-tyrosine} + 4\text{-methyl-2-oxopentanoate} \leftrightarrow 4\text{-hydroxyphenylpyruvate} + \text{L-leucine} \]
araT_3  L-tyrosine + 3-methyl-2-oxopentanoate <=> 4-hydroxyphenylpyruvate + L-isoleucine
araT_4  L-tyrosine + phenylpyruvate <=> 4-hydroxyphenylpyruvate + L-phenylalanine
araT_5  L-tyrosine + 4-methylthio 2-oxobutyrate <=> 4-hydroxyphenylpyruvate + L-methionine
araT_6  L-tyrosine + indol-3-pyruvate <=> 4-hydroxyphenylpyruvate + L-tryptophan
araT_8  L-phenylalanine + 4-methyl-2-oxopentanoate <=> phenylpyruvate + L-leucine
araT_9  L-phenylalanine + 3-methyl-2-oxopentanoate <=> phenylpyruvate + L-isoleucine
bPOLYS  5.5 UDP-glucose + UDP-galactose + 5.6 dTDP-rhamnose => POLYS + 6.5 UDP + 5.6 dTDP
c140d_1  tetradecanoate => diphosphate
        Acetyl-(acp) + 6 Malonyl-(acp) + 12 NADP + 12 H(+)
        tetradecanoyl-(acp) + 12 NADP+ + 6 CO(2) + 6 ACP + 6 H(2)O
        Acetyl-(acp) + 6 Malonyl-(acp) + 11 NADP + 11 H(+)
        hexadecanoate + ATP + 8 CoA + 8 FAD + 8 NAD + 8 H(+) => AMP + Acetyl-CoA
        Acetyl-(acp) + 7 Malonyl-(acp) + 14 NADP + 14 H(+)
        11 NADP(+) + 6 Co(2) + 6 ACP + 6 H(2)O
        Acetyl-(acp) + 7 Malonyl-(acp) + 13 NADP + 13 H(+)
        octadecanoate + ATP + 9 CoA + 9 FAD + 9 NAD + 9 H(+) => AMP + Acetyl-CoA
        Acetyl-(acp) + 8 Malonyl-(acp) + 16 NADP + 16 H(+)
        octadecanoyl-(acp) + 16 NADP+ + 8 CO(2) + 8 ACP + 8 H(2)O
        Acetyl-(acp) + 8 Malonyl-(acp) + 15 NADP + 15 H(+)
        octadecenoyl-(acp) + 15 NADP+ + 8 CO(2) + 8 ACP + 8 H(2)O
CDPk_1  ATP + CDP <=> ADP + CTP
CDPk_2  ATP + dCDP <=> ADP + dCTP
chem_1  2-acetolactate + O(2) => 2 diacetyl + 2 CO(2) + 2 H(2)O
chem_2  L-glutamate 5-semialdehyde <=> 1-pyrroline-5-carboxylate + H(2)O
chem_3  L-cysteine => cysteamine + CO(2)
d190s_1  octadecenoyl-(acp) => 11,12-methylene-octadecanoyl-(acp)
GDPk_2  dGDP + ATP <=> dGTP + ADP
hcoco_1  HCO(3) + H(+) => CO(2) + H(2)O
lacG_1  lactose 6-phosphate + H(2)O => D-glucose + D-galactose 6-phosphate
menA_1  1,4-dihydroxy-2-naphthoate + trans-heptaprenyl diphosphate => 2-
demethylmenaquinone + diphosphate + CO(2)

2-succinyl-6-hydroxy-2,4-cyclohexadiene-1-carboxylate => 2-succinylbenzoate + H(2)O

menC_1

0.5 O(2) + NADH + H(+) => H(2)O + NAD(+)

noxC_1

0.5 O(2) + NADH + H(+) => H(2)O + NAD(+)

noxD_1

0.5 O(2) + NADH + H(+) => H(2)O + NAD(+)

noxE_1

NAD(+) + ATP => NADP(+) + ADP

ppnK_1

4 isopentenyl diphosphate + trans-trans-farnesyl diphosphate => trans-heptaprenyl diphosphate + 4 diphosphate

preA_1

5-amino-6-(5-phosphoribitylamino)uracil + 4-(1-D-ribitylamino)-5-amino-2,6-dihydroxypyrimidine => 2 6,7-dimethyl-8-(1-D-ribityl)umazine + phosphate

ribB_1

ATP + dTDP <=> ADP + dTTP

trePP_1

trehalose 6-phosphate + phosphate <=> D-glucose 6-phosphate + beta-D-glucose 1-phosphate

UDPk_1

ATP + UDP <=> ADP + UTP

unk_1

D-glucono-1,5-lactone 6-phosphate + H(2)O => 6-phospho-D-gluconate

unk_20

S-adenosyl-L-homocysteine + H(2)O <=> adenosine + L-homocysteine

unk_23

4-methylthio 2-oxobutyrate => methional + CO(2)

L-methionine + 4-methyl-2-oxopentanoate <=> 4-methylthio 2-oxobutyrate + L-leucine

unk_24

L-methionine + 3-methyl-2-oxopentanoate <=> 4-methylthio 2-oxobutyrate + L-isoleucine

unk_25

L-methionine + phenylpyruvate <=> 4-methylthio 2-oxobutyrate + L-phenylalanine

unk_26

L-methionine + 4-hydroxyphenylpyruvate <=> 4-methylthio 2-oxobutyrate + L-tyrosine

unk_27

L-methionine + indol-3-pyruvate <=> 4-methylthio 2-oxobutyrate + L-tryptophan

unk_28

LL-2,6-diaminoheptanedioate <=> meso-2,6-diaminoheptanedioate

unk_32

4-hydroxyphenylpyruvate + 0.5 O(2) => 4-hydroxyphenylacetate + CO(2)

unk_34

phenylacetaldehyde + NAD(+) + H(2)O => phenylacetate + NADH + H(+) + indol-3-acetaldehyde + NAD(+) + H(2)O => indol-3-acetate + NADH + H(+) + indol-3-pyruvate + NADH + H(+) => indol-3-lactate + NAD(+) + undecaprenyl diphosphate + H(2)O => undecaprenyl phosphate + phosphate

unk_40

undecaprenyl diphosphate + H(2)O => diacylglycerol 3-phosphate + H(2)O => diacylglycerol + phosphate
lyso phosphatidylglycerol 1-phosphate + H(2)O => lysophosphatidylglycerol + phosphate

UDP-glucose + diacylglycerol => UDP + monoglucosyl diacylglycerol

UDP-glucose + monoglucosyl diacylglycerol => UDP + diglucosyl diacylglycerol

2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine triphosphate + 3 H(2)O => 2-amino-4-hydroxy-6-(erythro-1,2,3-trihydroxypropyl)dihydropteridine + 3 phosphate

UDP-glucose + diacylglycerol => UDP + monoglucosyl diacylglycerol

3-phosphoadenylyl sulfate + reduced thioredoxin => oxidized thioredoxin + H(2)SO(3) + adenosine 3',5'-bisphosphate

(N)-malate + NAD(+) <=> oxaloacetate + NADH + H(+)

Acetyls-CoA => Acetyl-CoA

Acetyl-CoA + 3 NADPH + 3 H(+) => H(2)S + 3 NADP(+) + 3 H(2)O

3MOBextI 3MOBextX => 3MOBext
3MOBextO 3MOBext => 3MOBextX
3MOPextI 3MOPextX => 3MOPext
3MOPextO 3MOPext => 3MOPextX
4MOPextI 4MOPextX => 4MOPext
4MOPextO 4MOPext => 4MOPextX
AALDextI AALDextX => AALDext
AALDextO AALDext => AALDextX
AcCoAextO Acetyl-CoA => AcCoAextX
ACESextI ACESextX => ACESext
ACESextO ACESext => ACESextX
ACTEXTextI ACTTEXTextX => ACTTEXText
ACTEXTextO ACTTEXText => ACTTEXTextX
ACTNEXTextI ACTNEXTextX => ACTNEXText
ACTNEXTextO ACTNEXText => ACTNEXTextX
ADEextI ADEextX => ADEext
ADEextO ADEext => ADEextX
ALAextI ALAextX => ALAext
ALAextO ALAext => ALAextX
ALCTTextI ALCTTextX => ALCTText

287
ALCTextO  ALCText => ALCTextX

amtB_1  NH(3)ext <= NH(3)

ARBTextI  ARBTextX => ARBText

ARBTextO  ARBText => ARBTextX

ARGextI  ARGextX => ARGext

ARGextO  ARGext => ARGextX

ASNextI  ASNextX => ASNext

ASNextO  ASNext => ASNextX

ASPextI  ASPextX => ASPext

ASPextO  ASPext => ASPextX

BDOHextI  BDOHextX => BDOHext

BDOHextO  BDOHext => BDOHextX

BIOTextI  BIOTextX => BIOText

BIOTextO  BIOText => BIOTextX

C120extI  C120extX => C120ext

C120extO  C120ext => C120extX

C140extI  C140extX => C140ext

C140extO  C140ext => C140extX

C160extI  C160extX => C160ext

C160extO  C160ext => C160extX

C160_2extI  C160_2extX => C160_2ext

C160_2extO  C160_2ext => C160_2extX

C180extI  C180extX => C180ext

C180extO  C180ext => C180extX

C180_2extI  C180_2extX => C180_2ext

C180_2extO  C180_2ext => C180_2extX

CELBextI  CELBextX => CELBext

CELBextO  CELBext => CELBextX

CHOextI  CHOextX => CHOext

CHOextO  CHOext => CHOextX

choS_1  CHOext + ATP + H(2)O => choline + ADP + phosphate

CO2extI  CO(2)extX => CO(2)ext

CO2extO  CO(2)ext => CO(2)extX
CYSextI  CYSextX => CYSext
CYSextO  CYSext => CYSextX
CYSTextI  CYSTextX => CYSText
CYSTextO  CYSText => CYSTextX
DPEPextI  DPEPextX => DPEPext
DPEPextO  DPEPext => DPEPextX
DTYLextI  DTYLextX => DTYLext
DTYLextO  DTYLext => DTYLextX
ETOHexI  ETOHexX => ETOHex
ETOHexO  ETOHex => ETOHexX
FOLextI  FOLextX => FOLext
FOLextO  FOLext => FOLextX
FORMextI  FORMextX => FORMext
FORMextO  FORMext => FORMextX
FRUTextI  FRUTextX => FRUText
FRUTextO  FRUText => FRUTextX
GABAextI  GABAextX => GABAext
GABAextO  GABAext => GABAextX
GLCextI  GLCextX => GLCext
GLCextO  GLCext => GLCextX
GLCNextI  GLCNextX => GLCNext
GLCNextO  GLCNext => GLCNextX
GLNextI  GLNextX => GLNext
GLNextO  GLNext => GLNextX
GLNTextI  GLNTextX => GLNText
GLNTextO  GLNText => GLNTextX
GALNTextI  GALNTextX => GALNText
GALNTextO  GALNText => GALNTextX
glpF1_1  GLYCext <=> glycerol
glpF2_1  GLYCext <=> glycerol
GLUextI  GLUextX => GLUext
GLUextO  GLUext => GLUextX
GLUexti  GLUextX => GLUext

289
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Reaction Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLUextO</td>
<td>GLUext =&gt; GLUextX</td>
</tr>
<tr>
<td>GLUMextI</td>
<td>GLUMextX =&gt; GLUMext</td>
</tr>
<tr>
<td>GLUMextO</td>
<td>GLUMext =&gt; GLUMextX</td>
</tr>
<tr>
<td>GLYC3PextI</td>
<td>GLYC-3-PextX =&gt; GLYC-3-PextX</td>
</tr>
<tr>
<td>GLYC3PextO</td>
<td>GLYC-3-Pext =&gt; GLYC-3-PextX</td>
</tr>
<tr>
<td>GLYCextI</td>
<td>GLYCextX =&gt; GLYCext</td>
</tr>
<tr>
<td>GLYCextO</td>
<td>GLYCext =&gt; GLYCextX</td>
</tr>
<tr>
<td>GLYextI</td>
<td>GLYextX =&gt; GLYext</td>
</tr>
<tr>
<td>GLYextO</td>
<td>GLYext =&gt; GLYextX</td>
</tr>
<tr>
<td>GNNextI</td>
<td>GNNextX =&gt; GNNext</td>
</tr>
<tr>
<td>GNNextO</td>
<td>GNNext =&gt; GNNextX</td>
</tr>
<tr>
<td>HextI</td>
<td>H(+)extX =&gt; H(+)ext</td>
</tr>
<tr>
<td>HextO</td>
<td>H(+)ext =&gt; H(+)extX</td>
</tr>
<tr>
<td>H2OextI</td>
<td>H(2)OextX =&gt; H(2)Oext</td>
</tr>
<tr>
<td>H2OextO</td>
<td>H(2)Oext =&gt; H(2)OextX</td>
</tr>
<tr>
<td>HISextI</td>
<td>HISextX =&gt; HISext</td>
</tr>
<tr>
<td>HISextO</td>
<td>HISext =&gt; HISextX</td>
</tr>
<tr>
<td>HPAextI</td>
<td>HPAextX =&gt; HPAext</td>
</tr>
<tr>
<td>HPAextO</td>
<td>HPAext =&gt; HPAextX</td>
</tr>
<tr>
<td>HPLextI</td>
<td>HPLextX =&gt; HPLext</td>
</tr>
<tr>
<td>HPLextO</td>
<td>HPLext =&gt; HPLextX</td>
</tr>
<tr>
<td>HYXNextI</td>
<td>HYXNextX =&gt; HYXNext</td>
</tr>
<tr>
<td>HYXNextO</td>
<td>HYXNext =&gt; HYXNextX</td>
</tr>
<tr>
<td>I3AextI</td>
<td>I3AextX =&gt; I3Aext</td>
</tr>
<tr>
<td>I3AextO</td>
<td>I3Aext =&gt; I3AextX</td>
</tr>
<tr>
<td>I3LextI</td>
<td>I3LextX =&gt; I3Lext</td>
</tr>
<tr>
<td>I3LextO</td>
<td>I3Lext =&gt; I3LextX</td>
</tr>
<tr>
<td>ILEextI</td>
<td>ILEextX =&gt; ILEext</td>
</tr>
<tr>
<td>ILEextO</td>
<td>ILEext =&gt; ILEextX</td>
</tr>
<tr>
<td>LACTextI</td>
<td>LACTextX =&gt; LACText</td>
</tr>
<tr>
<td>LACTextO</td>
<td>LACText =&gt; LACTextX</td>
</tr>
<tr>
<td>LCTTextI</td>
<td>LCTTextX =&gt; LCTText</td>
</tr>
<tr>
<td>LCTTextO</td>
<td>LCTText =&gt; LCTTextX</td>
</tr>
</tbody>
</table>

290
| LEUextI | LEUextX => LEUext |
| LEUextO | LEUext => LEUextX |
| LYSextI | LYSextX => LYSext |
| LYSextO | LYSext => LYSextX |
| MALAextI | MALAextX => MALAext |
| MALAextO | MALAext => MALAextX |
| MALTextI | MALTextX => MALText |
| MALTextO | MALText => MALTextX |
| MANNextI | MANNextX => MANNext |
| MANNextO | MANNext => MANNextX |
| METextI | METextX => METext |
| METextO | METext => METextX |
| MNTLextI | MNTLextX => MNTLext |
| MNTLextO | MNTLext => MNTLextX |
| MTALextI | MTALextX => MTALext |
| MTALextO | MTALext => MTALextX |
| MTTLextI | MTTLextX => MTTLext |
| MTTLextO | MTTLext => MTTLextX |
| NaextI | Na(+)-extX => Na(+)-ext |
| NaextO | Na(+)-ext => Na(+)-extX |
| NH3extI | NH(3)-extX => NH(3)-ext |
| NH3extO | NH(3)-ext => NH(3)-extX |
| NICOextI | NICOextX => NICOext |
| NICOextO | NICOext => NICOextX |
| O2extI | O(2)-extX => O(2)-ext |
| O2extO | O(2)-ext => O(2)-extX |
| OPEPextI | OPEPextX => OPEPext |
| OPEPextO | OPEPext => OPEPextX |
| ORNextI | ORNextX => ORNext |
| ORNextO | ORNext => ORNextX |
| PACTextI | PACTextX => PACText |
| PACTextO | PACText => PACTextX |
| PANextI | PANextX => PANext |
| PANextO | PANext => PANextX |
SUCRextO \rightarrow SUCRextX
THIextI \rightarrow THIext
THIextO \rightarrow THIextX
THRextI \rightarrow THRext
THRextO \rightarrow THRextX
TPEPextI \rightarrow TPEPextX
TPEPextO \rightarrow TPEPextX
TRHLextI \rightarrow TRHLext
TRHLextO \rightarrow TRHLextX
TRPextI \rightarrow TRPext
TRPextO \rightarrow TRPextX
TYRextI \rightarrow TYRext
TYRextO \rightarrow TYRextX
URAextI \rightarrow URAextX
URAextO \rightarrow URAextX
VALextI \rightarrow VALext
VALextO \rightarrow VALextX
XANextI \rightarrow XANext
XANextO \rightarrow XANextX
XYLANextI \rightarrow XYLANext
XYLANextO \rightarrow XYLANextX
XYLexI \rightarrow XYLex
XYLexO \rightarrow XYLexX

\text{BIOMass} = 4.201 \text{ PROT} + 0.074 \text{ DNA} + 0.329 \text{ RNA} + 0.119 \text{ PG} + 0.064 \text{ POLYS} + 18.15 \text{ ATP} + 0.015 \text{ LTA} + 0.032 \text{ LIP} \rightarrow \text{BIOMASS} + 18.15 \text{ ADP} + 18.15 \text{ phosphate}

\text{intimin} = 23 \text{ glycine} + 27 \text{ L-alanine} + 27 \text{ L-valine} + 13 \text{ L-leucine} + 16 \text{ L-isoleucine} + 30 \text{ L-serine} + 2 \text{ L-cysteine} + 34 \text{ L-threonine} + 1 \text{ L-methionine} + 6 \text{ L-proline} + 6 \text{ L-phenylalanine} + 11 \text{ L-tyrosine} + 4 \text{ L-tryptophan} + 29 \text{ L-lysine} + 2 \text{ L-arginine} + 14 \text{ L-aspartate} + 9 \text{ L-glutamate} + 21 \text{ L-asparagine} + 11 \text{ L-glutamine} + 1231.52 \text{ ATP} + 944.516 \text{ H}_2\text{O} \rightarrow \text{Intimin} + 1231.52 \text{ ADP} + 1231.52 \text{ phosphate}

\text{LIPass} = 18.9 \text{ phosphatidylycerol} + 42.5 \text{ cardiolipin} + 30.3 \text{ diglucosyl diacylglycerol} + 4 \text{ monoglucosyl diacylglycerol} + 4.3 \text{ lysophosphatidylglycerol} \rightarrow 100 \text{ LIP}

\text{lump}_1 = N\text{-acetyl muramoyl-} (N\text{-acetyl-D-glucosamine}) - \text{L-alanyl-D-glutamyl-L-lysyl-D-alanyl-D-alanine-diphosphoylcalprenol} + \text{D-aspartate} \rightarrow \text{L-alanine} + \text{undecaprenyl diphosphate} + \text{PG}

\text{lump}_2 = 15 \text{ phosphatidylycerol} + \text{glycerol 3-phosphate} \rightarrow 16 \text{ glycerol}

\text{lump}_3 = 6.08 \text{ D-alanine} + 9.76 \text{ UDP-galactose} + 16 \text{ glycerol phosphate} \rightarrow \text{D-alanyl-D-galactosyl-}
poly(glycerol phosphate) + 9.76 UDP + 6.08 H(2)O

D-alanyl-D-galactosyl-poly(glycerol phosphate) + diglucosyl

lump_4
diacetylglucose => LTA + H(2)O

8.6 L-alanine + 4.1 L-arginine + 3.1 L-aspartate + 5.9 L-asparagine + 3.4 L-cysteine + 3.6 L-glutamate + 6.4 L-glutamine + 9.2 glycine + 1.5 L-histidine + 6.1 L-isoleucine + 8.7 L-leucine + 7.2 L-lysine + 2.5 L-methionine + 3.8 L-phenylalanine + 3.5 L-proline + 5.1 L-serine + 5.6 L-threonine + 1.7 L-tryptophan + 2.7 L-tyrosine + 7.2 L-valine + 430.6 ATP

PROTass

=> 100 PROT + 430.6 phosphate + 430.6 ADP

26.2 AMP + 20 CMP + 32.2 GMP + 21.6 UMP + 240 ATP => 100 RNA + 240 ADP + 240 phosphate

RNAass
glycerol 3-phosphate + 0.2941 hexadecanoyl-(acp) + 0.0270 hexadecenoyl-(acp) + 0.0116 octadecanoyl-(acp) + 0.4359 octadecenoyl-(acp) + 0.1341 11,12-methylene-octadecanoyl-(acp) => ACP + acylglycerol 3-phosphate

trip_1
glycerol 3-phosphate + 0.1825 tetradecanoyl-(acp) + 0.0121 trans-tetradec-2-enoyl-(acp) + 0.5883 hexadecanoyl-(acp) + 0.0539 hexadecenoyl-(acp) + 0.0231 octadecanoyl-(acp) + 0.8718 octadecenoyl-(acp) + 0.2682 11,12-methylene-octadecanoyl-(acp) => 2 ACP + diacetylglucose 3-phosphate

unk_43
glycerol 3-phosphate + 0.233244 glycine + 0.219169 L-alanine + 0.17996 L-valine + 0.165885 L-leucine + 0.162869 L-isoleucine + 0.0552949 L-serine + 0.0482574 L-cysteine + 0.0492627 L-threonine + 0.036193 L-methionine + 0.313673 L-proline + 0.014611 L-phenylalanine + 0.0874665 L-tyrosine + 0.0814343 L-histidine + 0.0995308 L-lysine + 0.282507 L-asparagine + 0.591153 L-glutamate + 0.0904826 L-asparagine + 0.0180965 L-glycine

unk_44
glycerol 3-phosphate + 0.1825 tetradecanoyl-(acp) + 0.0121 trans-tetradec-2-enoyl-(acp) + 0.5883 hexadecanoyl-(acp) + 0.0539 hexadecenoyl-(acp) + 0.0231 octadecanoyl-(acp) + 0.8718 octadecenoyl-(acp) + 0.2682 11,12-methylene-octadecanoyl-(acp) => 2 ACP + diacetylglucose 3-phosphate

arcD1_1

ARGext + L-ornithine => L-arginine + ORNext

arcD2_1

ARGext + L-ornithine => L-arginine + ORNext

brnQ_1

LEUext + H(+)ext <=> L-leucine + H(+)

brnQ_2

ILEext + H(+)ext <=> L-isoleucine + H(+)

brnQ_3

VALext + H(+)ext <=> L-valine + H(+)

dip_1
dipeptide + H(2)O => 0.155496 glycine + 0.146113 L-alanine + 0.119973 L-valine + 0.11059 L-leucine + 0.108579 L-isoleucine + 0.0368633 L-serine + 0.0321716 L-cysteine + 0.0328418 L-threonine + 0.0241287 L-methionine + 0.209115 L-proline + 0.0764075 L-phenylalanine + 0.0201072 L-tyrosine + 0.058311 L-histidine + 0.0663539 L-lysine + 0.0938338 L-asparagine + 0.188338 L-aspartate + 0.394102 L-glutamate + 0.0603217 L-asparagine + 0.0120643 L-glycine

dNAass

32.3 dAMP + 17.7 dCMP + 17.7 dGMP + 32.3 dTMP + 337.2 ATP =>

100 DNA + 337.2 ADP + 337.2 phosphate

dtpP_1

dPEPext + ATP + H(2)O => dipeptide + ADP + phosphate

E4PextO

D-erythrose 4-phosphate => E4PextX

F6PextO

D-fructose 6-phosphate => F6PextX

G3PextO

D-glyceraldehyde 3-phosphate => G3PextX

G6PextO

D-glucose 6-phosphate => G6PextX
gadC_1  GLUext + 4-aminobutanoate => L-glutamate + GABAext

GlnP_1  GLNexit + ATP => L-glutamine + ADP + phosphate

glpT_1  GLYC-3-Phext <=> glycerol 3-phosphate

gltP_1  GLUext + ATP => L-glutamate + ADP + phosphate

gltS_1  GLUext + ATP => L-glutamate + ADP + phosphate

gltS_2  ARGext + ATP => L-arginine + ADP + phosphate

gntP_1  GLNCexit + H(+)ext => D-glucurate + H(+)

GPextO  glycerone phosphate => GPexitX

H2O_trans_1  H(2)Oext <=> H(2)O

heme_1  2 NADH + 4 H(+) + O(2) => 2 NAD(+) + 2 H(+)ext + 2 H(2)O

lacS_1  LACTexit + H(+)ext <=> lactose + H(+)

lysP_1  LYSexit + H(+)ext <=> L-lysine + H(+)

lysQ_1  LYSexit + H(+)ext <=> L-lysine + H(+)

malF_1  MALText + H(+)ext <=> maltose + H(+)

malG_1  MALText + H(+)ext <=> maltose + H(+)

mleP_1  MALAext <=> (S)-malate

mtlA_1  GLUCexit + phosphoenolpyruvate => D-glucose 6-phosphate + Pyruvate

mtlA_2  MNTLexit + phosphoenolpyruvate => D-mannitol 1-phosphate + Pyruvate

nah_1  Na(+)ext + H(+) + ADP + phosphate => Na(+) + H(+)ext + ATP + H(2)O

nah_2  Na(+) + H(+)ext + ATP + H(2)O => Na(+)ext + H(+) + ADP + phosphate

OAextO  oxaloacetate => OAexitX

OGextO  2-oxoglutarate => OGexitX

Opp  OPEPexit + ATP => oligopeptide + ADP + phosphate

p3MOB_1  3-methyl-2-oxobutanoate => 3MOBext

p3MOP_1  3-methyl-2-oxopentanoate => 3MOPext

p4MOP_1  4-methyl-2-oxopentanoate => 4MOPext

pAAALD_1  acetaldehyde <=> AALDext

pACET_1  acetate + H(+) <=> ACETexit + H(+)ext

pACTN_1  2-acetoin <=> ACTNext

pADE_1  ADEext + H(+)ext <=> adenine + H(+)

pALA_1  ALAext + H(+)ext <=> L-alanine + H(+)

295
pALCTT_1 2-acetolactate <=> ALCTText
pASN_1 ASNext + H(+)ext <=> L-asparagine + H(+)
pASP_1 ASPext + H(+)ext <=> L-aspartate + H(+)
pBDOH_1 (R,R)-butane-2,3-diol <=> BDOHext
pBIOT_1 BIOText + H(+)ext => biotin + H(+)
pbuX_1 XANext + H(+)ext <=> xanthine + H(+)
pC120_1 C120ext => dodecanoate
pC120_2 dodecanoate => C120ext
pC140_1 C140ext => tetradecanoate
pC140_2 tetradecanoate => C140ext
pC160_1 C160ext => hexadecanoate
pC160_2 hexadecanoate => C160ext
pC1602_1 C160_2ext => hexadecenoate
pC1602_2 hexadecenoate => C160_2ext
pC180_1 C180ext => octadecanoate
pC180_2 octadecanoate => C180ext
pCO2_1 CO(2) <=> CO(2)ext
pCYS_1 CYSext + H(+)ext <=> L-cysteine + H(+)
pCYST_1 CYSText + H(+)ext <=> cytosine + H(+)
pDTYL_1 diacetyl <=> DTYLext
pETOH_1 ethanol => ETOHext
pFOL_1 FOLext + H(+)ext => folate + H(+)
pFORM_1 formate <=> FORMext
pGABA_1 4-aminobutanoate => GABAext
pGALC_1 GALCext + H(+)ext <=> D-galactose + H(+)
pGLN L-glutamine => GLNnext
pGLUC_1 GLUCext + H(+)ext => D-glucose + H(+)
pGLY_1 GLYext + H(+)ext <=> glycine + H(+)
pGNN_1 GNNext + H(+)ext <=> guanine + H(+)
pHIS_1 HISext + H(+)ext <=> L-histidine + H(+)
pHPA_1 4-hydroxyphenylacetate => HPAext
pHPL_1 4-hydroxyphenyllactate => HPLext
pHYXN_1  HYXnext + H(+)_ext <=> hypoxanthine + H(+)
pI3A_1   indol-3-acetate => I3Aext
pI3L_1   indol-3-lactate => I3Lext
pLCTT_1  (S)-lactate + H(+) <=> LCTText + H(+)ext
pMET_1   METext + H(+)_ext <=> L-methionine + H(+)
pMTAL_1  methional => MTALext
pMTTL_1  methanethiol => MTTLext
pNICO_1  NICOext + H(+)ext => nicotinamide + H(+)
pO2_1    O(2)_ext <=> O(2)
potB_1   SPRMext + ATP => spermidine + ADP + phosphate
potB_2   PUTRext + ATP => putrescine + ADP + phosphate
potC_1   SPRMext + ATP => spermidine + ADP + phosphate
potC_2   PUTRext + ATP => putrescine + ADP + phosphate
pPACT_1  phenylacetate => PACText
pPAN_1   PANext + H(+)ext => pantothenate + H(+)
pPHE_1   PHEext + H(+)ext <=> L-phenylalanine + H(+)
pPLCT_1  phenyllactate => PLCText
pPRO_1   L-proline <=> PROext
pPYR_1   Pyruvate + H(+) => PYRext + H(+)ext
pPYRO_1  PYROext + H(+)ext => pyrodal + H(+)
pRIBF_1  RIBFext + H(+)ext => riboflavin + H(+)
pSER_1   SERext + H(+)ext <=> L-serine + H(+)
pstC_1   Piext + H(+)ext => phosphate + H(+)
pstC_2   PPiext + H(+)ext => diphosphate + H(+)
pSUC_1   succinate <=> SUCext
pSUCR_1  SUCRext + H(+)ext <=> sucrose + H(+)
ptGALC_1 GALCext + phosphoenolpyruvate => D-galactose 6-phosphate + Pyruvate
pTHI_1   THIext + H(+)ext => thiamine + H(+)
pTHR_1   THRext + H(+)ext <=> L-threonine + H(+)
pLACT_1  LACText + phosphoenolpyruvate => lactose 6-phosphate + Pyruvate
pTRP_1   TRPext + H(+)ext <=> L-tryptophan + H(+)
pTYR_1   TYRext + H(+)ext <=> L-tyrosine + H(+)
pyrP_1   URAext + H(+)ext <=> uracil + H(+)

297
Appendix 6b.

Simulations S1 to S6 performed using Flux Balance Analysis. LB = lower bounds, UB = Upper bounds.

<table>
<thead>
<tr>
<th>Gene code</th>
<th>L</th>
<th>B</th>
<th>UB</th>
<th>LB</th>
<th>UB</th>
<th>LB</th>
<th>UB</th>
<th>LB</th>
<th>UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>accA_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>accB_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>accD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ackA1_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ackA2_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acpD_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acpD_1_dup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acpS_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>acpS_1_dup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>add_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adhA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adhE_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adk_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>adk_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADPk_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agl_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agl_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aldB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>als_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyY_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyY_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyY_5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyY_6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyL_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyL_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyL_5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amyL_6</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ansB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>apt_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>araT_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>araT_10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>araT_11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>araT_12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
</tr>
<tr>
<td>arcA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arcC1_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arcC2_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>arcC3_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argC_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argD_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argE_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argF_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argG_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argH_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argI_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>argJ_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroC_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroD_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroE_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroF_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroH_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aroK_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asd_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>assB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>asnH_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aspC_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bacA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bcaT_1</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bcaT_2</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bcaT_3</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bglA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bglA_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bglA_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bPOLYS</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>butA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>butB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabF_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabG_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabZ_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabL_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabF_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabG_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabZ_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabL_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabF_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabG_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabZ_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabL_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabF_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabG_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FabZ_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FabI_4
FabF_5
FabG_5
FabZ_5
FabL_5
EF0365_1
FabF_6
FabG_6
FabZ_6
FabL_6
EF0365_2
FabF_7
FabG_7
FabZ_7
FabL_7
EF0365_3
desA1_1
EF0365_4
FabF_8
FabG_8
FabZ_8
FabL_8
EF0365_5
desA1_2
EF0365_6
c140d_1  0  0  0  0  0  0  0

c140s_1  0  0  0  0  0  0  0

c141s_1  0  0  0  0  0  0  0

c160d_1  0  0  0  0  0  0  0

c160s_1  0  0  0  0  0  0  0

c161s_1  0  0  0  0  0  0  0

c180d_1  0  0  0  0  0  0  0

c180s_1  0  0  0  0  0  0  0

302
c181s_1  0  0  0  0  0  0  0
(carA_1  0  0  0  0  0  0  0
(carB_1  0  0  0  0  0  0  0
(cb6p_1  
(cdd_1  
(cdd_2  0  0  0  0  0  0  0
(CDPk_1  
(CDPk_2  
(cdsA_1  
(cdsA_2  
(celB_1  0  0  0  0  0  0  0
(chem_1  0  0  0  0  0  0  0
(chem_2  
(chem_3  0  0  0  0  0  0  0
(citB_1  
(citB_2  0  0  0  0  0  0  0
(citB_3  0  0  0  0  0  0  0
(citE_1  
(clsA_1  
(clsB_1  
(cmk_1  
(cmk_2  
(coaA_1  0  0  0  0  0  0  0
(cpsM_1  0  0  0  0  0  0  0
(cysD_1  0  0  0  0  0  0  0
(cysD_2  0  0  0  0  0  0  0
(metX_1  0  0  0  0  0  0  0
(cysE_1  
(cysK_1  0  0  0  0  0  0  0
(cysM_1  0  0  0  0  0  0  0
(d190s_1  0  0  0  0  0  0  0
(dal_1  0  0  0  0  0  0  0
(dapA_1  0  0  0  0  0  0  0
<table>
<thead>
<tr>
<th>Gene</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dapB_1</td>
<td>0</td>
</tr>
<tr>
<td>dcdA_1</td>
<td>0</td>
</tr>
<tr>
<td>ddl_1</td>
<td>0</td>
</tr>
<tr>
<td>deoB_1</td>
<td></td>
</tr>
<tr>
<td>deoB_2</td>
<td></td>
</tr>
<tr>
<td>deoC_1</td>
<td></td>
</tr>
<tr>
<td>deoD_1</td>
<td>0</td>
</tr>
<tr>
<td>deoD_2</td>
<td></td>
</tr>
<tr>
<td>deoD_3</td>
<td></td>
</tr>
<tr>
<td>deoD_4</td>
<td></td>
</tr>
<tr>
<td>deoD_5</td>
<td>0</td>
</tr>
<tr>
<td>deoD_6</td>
<td></td>
</tr>
<tr>
<td>deoD_7</td>
<td></td>
</tr>
<tr>
<td>deoE_1</td>
<td></td>
</tr>
<tr>
<td>deoE_2</td>
<td></td>
</tr>
<tr>
<td>deoF_1</td>
<td></td>
</tr>
<tr>
<td>deoF_2</td>
<td></td>
</tr>
<tr>
<td>DIPHOS</td>
<td></td>
</tr>
<tr>
<td>dukA_1</td>
<td>0</td>
</tr>
<tr>
<td>dkeA_1</td>
<td></td>
</tr>
<tr>
<td>dukA_2</td>
<td>0</td>
</tr>
<tr>
<td>dkeA_2</td>
<td></td>
</tr>
<tr>
<td>dukB_1</td>
<td>0</td>
</tr>
<tr>
<td>dkeB_1</td>
<td></td>
</tr>
<tr>
<td>dukB_2</td>
<td>0</td>
</tr>
<tr>
<td>dkeB_2</td>
<td></td>
</tr>
<tr>
<td>dut_1</td>
<td>0</td>
</tr>
<tr>
<td>enoA_1</td>
<td></td>
</tr>
<tr>
<td>enoB_1</td>
<td></td>
</tr>
<tr>
<td>fabD_1</td>
<td></td>
</tr>
<tr>
<td>fabH_1</td>
<td></td>
</tr>
<tr>
<td>fadA_1</td>
<td></td>
</tr>
<tr>
<td>fbaA_1</td>
<td></td>
</tr>
<tr>
<td>fbp_1</td>
<td>0</td>
</tr>
<tr>
<td>femD_1</td>
<td></td>
</tr>
<tr>
<td>femD_2</td>
<td></td>
</tr>
<tr>
<td>fhs_1</td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>folB_1</td>
<td>0</td>
</tr>
<tr>
<td>folD_2</td>
<td>0</td>
</tr>
<tr>
<td>gadB_1</td>
<td>0</td>
</tr>
<tr>
<td>galM_1</td>
<td>0</td>
</tr>
<tr>
<td>GDPk_2</td>
<td>0</td>
</tr>
<tr>
<td>glmU_2</td>
<td>0</td>
</tr>
<tr>
<td>gltB_1</td>
<td>0</td>
</tr>
<tr>
<td>gmk_2</td>
<td>0</td>
</tr>
<tr>
<td>gpdA_1</td>
<td>0</td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>gpo_1</td>
<td>0</td>
</tr>
<tr>
<td>gshR_1</td>
<td>0</td>
</tr>
<tr>
<td>guaA_1</td>
<td>0</td>
</tr>
<tr>
<td>guaB_1</td>
<td>0</td>
</tr>
<tr>
<td>guaC_1</td>
<td>0</td>
</tr>
<tr>
<td>hasC_1</td>
<td></td>
</tr>
<tr>
<td>hcoco_1</td>
<td></td>
</tr>
<tr>
<td>hisA_1</td>
<td>0</td>
</tr>
<tr>
<td>hisB_1</td>
<td>0</td>
</tr>
<tr>
<td>hisC_1</td>
<td>0</td>
</tr>
<tr>
<td>hisD_1</td>
<td></td>
</tr>
<tr>
<td>hisD_2</td>
<td>0</td>
</tr>
<tr>
<td>hisD_3</td>
<td></td>
</tr>
<tr>
<td>hisF_1</td>
<td></td>
</tr>
<tr>
<td>hisG_1</td>
<td></td>
</tr>
<tr>
<td>hisH_1</td>
<td></td>
</tr>
<tr>
<td>hisI_1</td>
<td>0</td>
</tr>
<tr>
<td>hisI_2</td>
<td>0</td>
</tr>
<tr>
<td>hisK_1</td>
<td>0</td>
</tr>
<tr>
<td>hmcM_1</td>
<td>0</td>
</tr>
<tr>
<td>hom_1</td>
<td></td>
</tr>
<tr>
<td>hprT_1</td>
<td></td>
</tr>
<tr>
<td>hprT_2</td>
<td></td>
</tr>
<tr>
<td>hpt_1</td>
<td></td>
</tr>
<tr>
<td>hpt_2</td>
<td></td>
</tr>
<tr>
<td>icd_1</td>
<td>0</td>
</tr>
<tr>
<td>ilvA_1</td>
<td>0</td>
</tr>
<tr>
<td>ilvB_1</td>
<td>0</td>
</tr>
<tr>
<td>ilvB_2</td>
<td>0</td>
</tr>
<tr>
<td>ilvC_1</td>
<td>0</td>
</tr>
<tr>
<td>ilvC_2</td>
<td>0</td>
</tr>
<tr>
<td>ilvD_1</td>
<td>0</td>
</tr>
<tr>
<td>ilvD_2</td>
<td>0</td>
</tr>
</tbody>
</table>
| Gene     | Value 
|----------|-------
| ilvN_1   | 0     
| ipd_1    | 0     
| ispA_1   | 0     
| ispA_2   | 0     
| ispB_1   | 0     
| kdgA_1   | 0     
| kdgK_1   | 0     
| kdhB_1   | 0     
| lacC_1   | 0     
| lacG_1   | 0     
| lacZ_1   | 0     
| ldh_1    | 0     
| ldhB_1   | 0     
| ldhX_1   | 0     
| leuA_1   | 0     
| leuB_1   | 0     
| leuC_1   | 0     
| leuD_1   | 0     
| lysA_1   | 0     
| maa_1    | 0     
| mae_1    | 0     
| mapA_1   | 0     
| menA_1   | 0     
| menB_1   | 0     
| menC_1   | 0     
| menD_1   | 0     
| menE_1   | 0     
| menF_1   | 0     
| metA_1   | 0     
| metB1_1  | 0     
| metB2_1  | 0     
| metB1_2  | 0     
| metC_1   | 0     |
metE_1
metF_1  0  0  0  0  0  0  0
metK_1  0  0  0  0  0  0  0
mleS_1  0  0  0  0  0  0  0
mraY_1  0  0  0  0  0  0  0
mtdD_1
murA1_1  0  0  0  0  0  0  0
murA2_1  0  0  0  0  0  0  0
murB_1  0  0  0  0  0  0  0
murC_1  0  0  0  0  0  0  0
murD_1  0  0  0  0  0  0  0
murE_1  0  0  0  0  0  0  0
murF_1  0  0  0  0  0  0  0
murG_1  0  0  0  0  0  0  0
murH_1  0  0  0  0  0  0  0
mvaA_1  0  0  0  0  0  0  0
nadE_1  0  0  0  0  0  0  0
nagA_1
nagB_1
glgA
glgB
glgC
glgD
glgP_1
glgP_2
noxA_1  0  0  0  0  0  0  0
noxB_1  0  0  0  0  0  0  0
noxC_1  0  0  0  0  0  0  0
noxD_1  0  0  0  0  0  0  0
noxE_1  0  0  0  0  0  0  0
nrdD_1  0  0  0  0  0  0  0
nrdD_2  0  0  0  0  0  0  0
nrdD_3  0  0  0  0  0  0  0
<table>
<thead>
<tr>
<th>Gene</th>
<th>Value1</th>
<th>Value2</th>
<th>Value3</th>
<th>Value4</th>
<th>Value5</th>
<th>Value6</th>
<th>Value7</th>
</tr>
</thead>
<tbody>
<tr>
<td>nrdD_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_3_dup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdE_4_dup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_3_dup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nrdF_4_dup</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nucA_9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pabA_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pabB_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pabA_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pabB_2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>panE_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>panB_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pde_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pdhA_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>pdhB_1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdhC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdhD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdp_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdp_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdp_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pepV_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pdk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pfl_1</td>
<td>14.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgiA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgiA_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgk_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgmB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgxA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pgxA_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pheA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pmg_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pmi_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>poxl_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ppmK_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>preA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proA_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>proC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prsA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>prsB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pta_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purB_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purB_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purC_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purD_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purE_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Code</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purF_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purH_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purH_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purK_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purL_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purM_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purN_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>purQ_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pycA_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pycA_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pycA_3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pycA_4</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pydA_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pydB_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pydB_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pydB_3</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyk_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrB_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrC_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrE_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrF_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrG_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pyrH_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>racD_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rbsK_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribA_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribA_KEGG</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribB_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribC_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribC_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribG_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribG_2</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribH_1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ribH_KEGG</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rmlA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rmlB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rmlC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpe_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ptk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rpiA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>scrK_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sdaA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sdaB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>serA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>serB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>serC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sodA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tagD1_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tagD2_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TDPk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiD_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiE_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tenA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiD_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ynbD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thiM_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thrA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thrA_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thrB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thrC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>thyA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tkt_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tkt_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tmk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tpiA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trePP_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpE_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpF_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trpG_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trxB1_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>trxB2_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tyrA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ubiE_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udk_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udk_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udp_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>udp_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDPk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UDPk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_10</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_11</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_16</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_18</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_19</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_20</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_21</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_22</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_23</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_24</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_65</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_7</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>unk_8</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>upp_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uppS_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaC_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>uxaB_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xpt_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylM_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylM_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylM_3</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylM_4</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xylM_5</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>glk_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xynB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xynB_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ycbD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ychH_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ydiD_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yeaB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yeaB_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yeaG_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yeaH_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yebA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yebB_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yfiG_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Expression Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yggA_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yjF_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ylaE_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ylaE_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yrjL_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ywiC_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ywiC_2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zwf_1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3MOBextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3MOBextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3MOPextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3MOPextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4MOPextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4MOPextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AALDextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AALDextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AcCoAextO</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACETextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACETextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTNextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACTNextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADEextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADEextO</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gene</th>
<th>Expression Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALAextI</td>
<td>3 0.573 0.5 0.573 0.573 0.573 0.573 0.5 0.573 0.573</td>
</tr>
<tr>
<td>ALAextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
</tr>
<tr>
<td>ALCTTextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>ALCTTextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
</tr>
<tr>
<td>antB_1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>ARBTextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>ARBTextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
</tr>
<tr>
<td>ARGextI</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
</tr>
<tr>
<td>ARGextO</td>
<td>0 1000 0 1000 0 1000 0 1000 0 1000 0 1000</td>
</tr>
<tr>
<td></td>
<td>ExtI</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>DTYL</td>
<td>0</td>
</tr>
<tr>
<td>DTY</td>
<td>0</td>
</tr>
<tr>
<td>ETOH</td>
<td>0</td>
</tr>
<tr>
<td>ETO</td>
<td>0</td>
</tr>
<tr>
<td>FO</td>
<td>0</td>
</tr>
<tr>
<td>FOL</td>
<td>0</td>
</tr>
<tr>
<td>FORM</td>
<td>0</td>
</tr>
<tr>
<td>FORM</td>
<td>0</td>
</tr>
<tr>
<td>FRU</td>
<td>0</td>
</tr>
<tr>
<td>FRU</td>
<td>0</td>
</tr>
<tr>
<td>GABA</td>
<td>0</td>
</tr>
<tr>
<td>GABA</td>
<td>0</td>
</tr>
<tr>
<td>GALC</td>
<td>0</td>
</tr>
<tr>
<td>GALC</td>
<td>0</td>
</tr>
<tr>
<td>GLC</td>
<td>0</td>
</tr>
<tr>
<td>GLC</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>GLN</td>
<td>0</td>
</tr>
<tr>
<td>Amino Acid</td>
<td>I</td>
</tr>
<tr>
<td>------------</td>
<td>-----</td>
</tr>
<tr>
<td>GLYextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>GNNextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>H2OextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HIextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>HPAextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HPLextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>HYXNextI</td>
<td>0 50 0 50 0 50 0 50 0 50 0 50 0 50</td>
</tr>
<tr>
<td>I3AextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>I3LextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>ILEextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>LACTextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LCTTextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>LEUextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>LYSextI</td>
<td>0 100 0 100 0 100 0 100 0 100 0 100 0 100</td>
</tr>
<tr>
<td>MALAextI</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>ExtI</td>
</tr>
<tr>
<td>------------------</td>
<td>------</td>
</tr>
<tr>
<td>MALTextI</td>
<td>0</td>
</tr>
<tr>
<td>MALTextO</td>
<td>1000</td>
</tr>
<tr>
<td>MANNextI</td>
<td>0</td>
</tr>
<tr>
<td>MANNextO</td>
<td>1000</td>
</tr>
<tr>
<td>METextI</td>
<td>0</td>
</tr>
<tr>
<td>METextO</td>
<td>1000</td>
</tr>
<tr>
<td>MNTLextI</td>
<td>0</td>
</tr>
<tr>
<td>MNTLextO</td>
<td>1000</td>
</tr>
<tr>
<td>MTALextI</td>
<td>0</td>
</tr>
<tr>
<td>MTALextO</td>
<td>1000</td>
</tr>
<tr>
<td>MTTLextI</td>
<td>0</td>
</tr>
<tr>
<td>MTTLextO</td>
<td>1000</td>
</tr>
<tr>
<td>NnextI</td>
<td>0</td>
</tr>
<tr>
<td>NnextO</td>
<td>1000</td>
</tr>
<tr>
<td>NH3extI</td>
<td>0</td>
</tr>
<tr>
<td>NH3extO</td>
<td>1000</td>
</tr>
<tr>
<td>NiCOextI</td>
<td>0</td>
</tr>
<tr>
<td>NiCOextO</td>
<td>0</td>
</tr>
<tr>
<td>O2extI</td>
<td>0</td>
</tr>
<tr>
<td>O2extO</td>
<td>1000</td>
</tr>
<tr>
<td>OPEPextI</td>
<td>0</td>
</tr>
<tr>
<td>OPEPextO</td>
<td>0</td>
</tr>
<tr>
<td>ORNextI</td>
<td>0</td>
</tr>
<tr>
<td>ORNextO</td>
<td>1000</td>
</tr>
<tr>
<td>PACTextI</td>
<td>0</td>
</tr>
<tr>
<td>PACTextO</td>
<td>1000</td>
</tr>
<tr>
<td>PANextI</td>
<td>0</td>
</tr>
<tr>
<td>PANextO</td>
<td>0</td>
</tr>
<tr>
<td>PHEextI</td>
<td>0</td>
</tr>
<tr>
<td>PHEextO</td>
<td>1000</td>
</tr>
<tr>
<td>PiextI</td>
<td>0</td>
</tr>
<tr>
<td>PiextO</td>
<td>1000</td>
</tr>
<tr>
<td>PLCTextI</td>
<td>0</td>
</tr>
<tr>
<td>PLCTextO</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>---------</td>
<td>---</td>
</tr>
<tr>
<td>PLCTextO</td>
<td></td>
</tr>
<tr>
<td>PPhexI</td>
<td></td>
</tr>
<tr>
<td>PPexI</td>
<td></td>
</tr>
<tr>
<td>PRexI</td>
<td></td>
</tr>
<tr>
<td>PRextI</td>
<td></td>
</tr>
<tr>
<td>PUTRextI</td>
<td></td>
</tr>
<tr>
<td>PUTRextO</td>
<td></td>
</tr>
<tr>
<td>PYexI</td>
<td></td>
</tr>
<tr>
<td>PYexO</td>
<td></td>
</tr>
<tr>
<td>PYexO_d</td>
<td></td>
</tr>
<tr>
<td>PYRexI</td>
<td></td>
</tr>
<tr>
<td>PYROexI</td>
<td></td>
</tr>
<tr>
<td>PYROexO</td>
<td></td>
</tr>
<tr>
<td>RIBexI</td>
<td></td>
</tr>
<tr>
<td>RIBexO</td>
<td></td>
</tr>
<tr>
<td>RIBFexI</td>
<td></td>
</tr>
<tr>
<td>RIBFexO</td>
<td></td>
</tr>
<tr>
<td>SALCexI</td>
<td></td>
</tr>
<tr>
<td>SALCexO</td>
<td></td>
</tr>
<tr>
<td>SERexI</td>
<td></td>
</tr>
<tr>
<td>SERexO</td>
<td></td>
</tr>
<tr>
<td>SLFexI</td>
<td></td>
</tr>
<tr>
<td>SLFexO</td>
<td></td>
</tr>
<tr>
<td>SPRMexI</td>
<td></td>
</tr>
<tr>
<td>SPRMexO</td>
<td></td>
</tr>
<tr>
<td>SUCCexI</td>
<td></td>
</tr>
<tr>
<td>SUCCexO</td>
<td></td>
</tr>
<tr>
<td>SUCRexI</td>
<td></td>
</tr>
<tr>
<td>SUCRexO</td>
<td></td>
</tr>
<tr>
<td>THexI</td>
<td></td>
</tr>
<tr>
<td>THexO</td>
<td></td>
</tr>
<tr>
<td>THRexI</td>
<td></td>
</tr>
<tr>
<td>THRexO</td>
<td></td>
</tr>
<tr>
<td>TPEPexI</td>
<td></td>
</tr>
<tr>
<td>Compound</td>
<td>TPEPextO</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Compound</td>
<td>Value</td>
</tr>
<tr>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>atpD</td>
<td></td>
</tr>
<tr>
<td>atpE</td>
<td></td>
</tr>
<tr>
<td>atpF</td>
<td></td>
</tr>
<tr>
<td>atpG</td>
<td></td>
</tr>
<tr>
<td>atpH</td>
<td></td>
</tr>
<tr>
<td>ATPp_1</td>
<td></td>
</tr>
<tr>
<td>brnQ_1</td>
<td></td>
</tr>
<tr>
<td>brnQ_2</td>
<td></td>
</tr>
<tr>
<td>brnQ_3</td>
<td></td>
</tr>
<tr>
<td>dip_1</td>
<td>0</td>
</tr>
<tr>
<td>DNAass</td>
<td>0</td>
</tr>
<tr>
<td>dtpP_1</td>
<td>0</td>
</tr>
<tr>
<td>dtpP_2</td>
<td>0</td>
</tr>
<tr>
<td>dtpT_1</td>
<td>0</td>
</tr>
<tr>
<td>dtpT_2</td>
<td>0</td>
</tr>
<tr>
<td>E4PextO</td>
<td>0</td>
</tr>
<tr>
<td>F6PextO</td>
<td>0</td>
</tr>
<tr>
<td>fruA_1</td>
<td>0</td>
</tr>
<tr>
<td>G3PextO</td>
<td>0</td>
</tr>
<tr>
<td>G6PextO</td>
<td>0</td>
</tr>
<tr>
<td>gadC_1</td>
<td>0</td>
</tr>
<tr>
<td>glnP_1</td>
<td>0</td>
</tr>
<tr>
<td>glpT_1</td>
<td></td>
</tr>
<tr>
<td>glpP_1</td>
<td>0</td>
</tr>
<tr>
<td>glS_1</td>
<td>0</td>
</tr>
<tr>
<td>gltS_2</td>
<td>0</td>
</tr>
<tr>
<td>gntP_1</td>
<td></td>
</tr>
<tr>
<td>GextO</td>
<td>0</td>
</tr>
<tr>
<td>H2O_trans_1</td>
<td>0</td>
</tr>
<tr>
<td>heme_1</td>
<td></td>
</tr>
<tr>
<td>lacS_1</td>
<td></td>
</tr>
<tr>
<td>lysP_1</td>
<td></td>
</tr>
<tr>
<td>lysQ_1</td>
<td></td>
</tr>
<tr>
<td>Gene</td>
<td>Value 1</td>
</tr>
<tr>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>maeP_1</td>
<td>0</td>
</tr>
<tr>
<td>malF_1</td>
<td>0</td>
</tr>
<tr>
<td>malG_1</td>
<td>0</td>
</tr>
<tr>
<td>mleP_1</td>
<td>0</td>
</tr>
<tr>
<td>mltA_1</td>
<td>0</td>
</tr>
<tr>
<td>mltA_2</td>
<td>0</td>
</tr>
<tr>
<td>mltF_1</td>
<td>0</td>
</tr>
<tr>
<td>mltF_2</td>
<td>0</td>
</tr>
<tr>
<td>nah_1</td>
<td>0</td>
</tr>
<tr>
<td>nah_2</td>
<td>0</td>
</tr>
<tr>
<td>OAextO</td>
<td>0</td>
</tr>
<tr>
<td>OGextO</td>
<td>0</td>
</tr>
<tr>
<td>Opp</td>
<td>0</td>
</tr>
<tr>
<td>p3MOB_1</td>
<td>0</td>
</tr>
<tr>
<td>p3MOP_1</td>
<td>0</td>
</tr>
<tr>
<td>p4MOP_1</td>
<td>0</td>
</tr>
<tr>
<td>pAALD_1</td>
<td>0</td>
</tr>
<tr>
<td>pACET_1</td>
<td>0</td>
</tr>
<tr>
<td>pADE_1</td>
<td>0</td>
</tr>
<tr>
<td>pALA_1</td>
<td>0</td>
</tr>
<tr>
<td>pALCTT_1</td>
<td>0</td>
</tr>
<tr>
<td>pASN_1</td>
<td>0</td>
</tr>
<tr>
<td>pASP_1</td>
<td>0</td>
</tr>
<tr>
<td>pBDOH_1</td>
<td>0</td>
</tr>
<tr>
<td>pBIOT_1</td>
<td>0</td>
</tr>
<tr>
<td>pbuX_1</td>
<td>0</td>
</tr>
<tr>
<td>pC120_1</td>
<td>0</td>
</tr>
<tr>
<td>pC120_2</td>
<td>0</td>
</tr>
<tr>
<td>pC140_1</td>
<td>0</td>
</tr>
<tr>
<td>pC140_2</td>
<td>0</td>
</tr>
<tr>
<td>pC160_1</td>
<td>0</td>
</tr>
<tr>
<td>pC160_2</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>---------------------</td>
<td>---</td>
</tr>
<tr>
<td>pC1602_1</td>
<td></td>
</tr>
<tr>
<td>pC1602_2</td>
<td></td>
</tr>
<tr>
<td>pC180_1</td>
<td></td>
</tr>
<tr>
<td>pC180_2</td>
<td></td>
</tr>
<tr>
<td>pCO2_1</td>
<td></td>
</tr>
<tr>
<td>pCYS_1</td>
<td></td>
</tr>
<tr>
<td>pCYST_1</td>
<td></td>
</tr>
<tr>
<td>pDTYL_1</td>
<td></td>
</tr>
<tr>
<td>PEPextO</td>
<td>0</td>
</tr>
<tr>
<td>pETOH_1</td>
<td>0</td>
</tr>
<tr>
<td>pFOL_1</td>
<td></td>
</tr>
<tr>
<td>pFORM_1</td>
<td></td>
</tr>
<tr>
<td>pGABA_1</td>
<td>0</td>
</tr>
<tr>
<td>pGALC_1</td>
<td></td>
</tr>
<tr>
<td>pGLN</td>
<td>0</td>
</tr>
<tr>
<td>pGLUC_1</td>
<td>0</td>
</tr>
<tr>
<td>pGLY_1</td>
<td></td>
</tr>
<tr>
<td>pGNN_1</td>
<td></td>
</tr>
<tr>
<td>pHIS_1</td>
<td></td>
</tr>
<tr>
<td>pHPA_1</td>
<td>0</td>
</tr>
<tr>
<td>pHPL_1</td>
<td>0</td>
</tr>
<tr>
<td>pHYXN_1</td>
<td></td>
</tr>
<tr>
<td>pI3A_1</td>
<td>0</td>
</tr>
<tr>
<td>pI3L_1</td>
<td>0</td>
</tr>
<tr>
<td>pLCTT_1</td>
<td></td>
</tr>
<tr>
<td>pMET_1</td>
<td></td>
</tr>
<tr>
<td>pMTAL_1</td>
<td>0</td>
</tr>
<tr>
<td>pMTTL_1</td>
<td>0</td>
</tr>
<tr>
<td>pNICO_1</td>
<td>0</td>
</tr>
<tr>
<td>pO2_1</td>
<td></td>
</tr>
<tr>
<td>potB_1</td>
<td>0</td>
</tr>
<tr>
<td>potB_2</td>
<td>0</td>
</tr>
<tr>
<td>potC_1</td>
<td>0</td>
</tr>
<tr>
<td>Gene</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>potC_2</td>
<td></td>
</tr>
<tr>
<td>pPACT_1</td>
<td></td>
</tr>
<tr>
<td>pPAN_1</td>
<td></td>
</tr>
<tr>
<td>pPHE_1</td>
<td></td>
</tr>
<tr>
<td>pPLCT_1</td>
<td></td>
</tr>
<tr>
<td>pPRO_1</td>
<td></td>
</tr>
<tr>
<td>pPYR_1</td>
<td></td>
</tr>
<tr>
<td>pPYRO_1</td>
<td></td>
</tr>
<tr>
<td>pRIBF_1</td>
<td></td>
</tr>
<tr>
<td>pSER_1</td>
<td></td>
</tr>
<tr>
<td>pstC_1</td>
<td></td>
</tr>
<tr>
<td>pstC_2</td>
<td></td>
</tr>
<tr>
<td>pstC_3</td>
<td></td>
</tr>
<tr>
<td>pstC_4</td>
<td></td>
</tr>
<tr>
<td>pstC_5</td>
<td></td>
</tr>
<tr>
<td>pSUC_1</td>
<td></td>
</tr>
<tr>
<td>pSUCR_1</td>
<td></td>
</tr>
<tr>
<td>ptbA_2</td>
<td></td>
</tr>
<tr>
<td>ptbA_3</td>
<td></td>
</tr>
<tr>
<td>ptbA_4</td>
<td></td>
</tr>
<tr>
<td>ptcA_1</td>
<td></td>
</tr>
<tr>
<td>ptcA_2</td>
<td></td>
</tr>
<tr>
<td>ptcA_3</td>
<td></td>
</tr>
<tr>
<td>ptcB_1</td>
<td></td>
</tr>
<tr>
<td>ptcB_2</td>
<td></td>
</tr>
<tr>
<td>ptcB_3</td>
<td></td>
</tr>
<tr>
<td>ptcB_4</td>
<td></td>
</tr>
<tr>
<td>ptcB_5</td>
<td></td>
</tr>
<tr>
<td>ptcC_1</td>
<td></td>
</tr>
<tr>
<td>ptcC_2</td>
<td></td>
</tr>
<tr>
<td>ptcC_3</td>
<td></td>
</tr>
<tr>
<td>ptcC_4</td>
<td></td>
</tr>
<tr>
<td>ptcC_5</td>
<td></td>
</tr>
<tr>
<td>ptGALC_1</td>
<td></td>
</tr>
<tr>
<td>pTHI_1</td>
<td></td>
</tr>
<tr>
<td>pTHI_2</td>
<td></td>
</tr>
<tr>
<td>pTHI_3</td>
<td></td>
</tr>
<tr>
<td>pTHI_4</td>
<td></td>
</tr>
<tr>
<td>pTHI_5</td>
<td></td>
</tr>
<tr>
<td>ptnAB_1</td>
<td></td>
</tr>
<tr>
<td>ptnAB_2</td>
<td></td>
</tr>
<tr>
<td>ptnAB_3</td>
<td></td>
</tr>
<tr>
<td>ptnAB_4</td>
<td></td>
</tr>
<tr>
<td>ptnAB_5</td>
<td></td>
</tr>
<tr>
<td>ptnC_1</td>
<td></td>
</tr>
<tr>
<td>ptnC_2</td>
<td></td>
</tr>
<tr>
<td>ptnC_3</td>
<td></td>
</tr>
<tr>
<td>ptnC_4</td>
<td></td>
</tr>
<tr>
<td>ptnC_5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>----------</td>
<td>-----</td>
</tr>
<tr>
<td>ptaD_1</td>
<td></td>
</tr>
<tr>
<td>ptaD_2</td>
<td></td>
</tr>
<tr>
<td>ptaD_3</td>
<td></td>
</tr>
<tr>
<td>ptaD_4</td>
<td></td>
</tr>
<tr>
<td>ptaD_5</td>
<td></td>
</tr>
<tr>
<td>pTRP_1</td>
<td></td>
</tr>
<tr>
<td>pTYR_1</td>
<td></td>
</tr>
<tr>
<td>pyrP_1</td>
<td></td>
</tr>
<tr>
<td>R5PextO</td>
<td></td>
</tr>
<tr>
<td>rbsC_1</td>
<td></td>
</tr>
<tr>
<td>rbsD_1</td>
<td></td>
</tr>
<tr>
<td>sucCoAextO</td>
<td>0</td>
</tr>
<tr>
<td>uxuT_1</td>
<td></td>
</tr>
<tr>
<td>exuT_1</td>
<td></td>
</tr>
<tr>
<td>xylT_1</td>
<td></td>
</tr>
<tr>
<td>xynT_1</td>
<td></td>
</tr>
<tr>
<td>xynT_2</td>
<td></td>
</tr>
<tr>
<td>yafB_1</td>
<td></td>
</tr>
<tr>
<td>yleD_1</td>
<td></td>
</tr>
<tr>
<td>yleD_2</td>
<td></td>
</tr>
<tr>
<td>lta_c_e</td>
<td></td>
</tr>
<tr>
<td>lta_e_b</td>
<td></td>
</tr>
<tr>
<td>lip_c_e</td>
<td></td>
</tr>
<tr>
<td>lip_e_b</td>
<td></td>
</tr>
<tr>
<td>BIOMASS_c_e</td>
<td>0</td>
</tr>
<tr>
<td>BIOMASS_e_b</td>
<td>0</td>
</tr>
<tr>
<td>MENAQUIN OL_c_e</td>
<td>0</td>
</tr>
<tr>
<td>MENAQUIN OL_e_b</td>
<td>0</td>
</tr>
<tr>
<td>intimin_1</td>
<td>1000</td>
</tr>
<tr>
<td>intimin_2</td>
<td></td>
</tr>
</tbody>
</table>

327
Appendix 6c.

MATLAB code for the function install_blast() of M2N Toolbox.

```matlab
function [result_status] = install_blast()
%
% This function has been developed to install BLAST locally.
% install_blast verifies if BLAST tools are already installed and, if
% not, it install BLAST from the internet or from a local file pointed by
% the user.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Check if there is a version of BLAST already installed and working.
result_status = 0;
disp('Checking if the tools blastp and blastdbcmd are installed and working properly...');

% blastp
blastpworking = system('blastp -help');
if blastpworking == 0
    disp('blastp is working.');
end

% blastdbcmd
blastdbcmdworking = system('blastdbcmd -help');
if blastdbcmdworking == 0
    disp('blastdbcmd is working.');
end

% If BLAST is not working, ask user if he wants to install BLAST.
if blastpworking == 0 && blastdbcmdworking == 0
    disp('You have no need to install BLAST. Your current installation is working fine.');
    decision = 2;
    result_status = 1;
else
    % blast installation is not working.
    % Create a dialog box asking users if they want to check their
    % installation, manually, or if they want to continue and try to
    % install it.
    % Construct dialog box
    disp('Your BLAST installation is not working. We are now looking for the blastinst.log,
which should be in your HOME folder.');
    blastinstfile = exist(fullfile(char(getenv('HOME')),'blastinst.log'), 'file');
    if blastinstfile == 2
        disp('File found in the HOME directory.);
        disp('Loading blastinst.log...');
        blastinst = loadcell(fullfile(char(getenv('HOME')),'blastinst.log'),',','');
        disp('blastinst.log loaded');
        binfo = blastinst(2);
        choice = questdlg(strcat('The bin folder is defined as -> ', binfo, '. We will try
to fix the problem with BLAST by setting the bin folder as part of your $PATH. Press OK to
confirm or press QUIT to stop the function.'), 'BLAST installation', 'OK', 'QUIT', 'OK');
        switch choice
            case 'OK'
                decision1 = 0;
            case 'QUIT'
                decision1 = 1;
        end
        if decision1 == 0
            oldpath = getenv('PATH');
            envpath = strcat(oldpath, ' :', binfo);
            disp('Setting PATH variable...');
            setenv('PATH', char(envpath));
            disp('PATH variable updated to:');
            disp(cellstr(envpath));
            % blastp
```
blastpworking = system('blastp -help');
if blastpworking == 0
    disp('blastp is now working.');</end

% blastdbcmd
blastdbcmdworking = system('blastdbcmd -help');
if blastdbcmdworking == 0
    disp('blastdbcmd is now working.');</end

% If BLAST is not working, ask user if he wants to install BLAST.
if blastpworking == 0 && blastdbcmdworking == 0
    disp('Your BLAST installation is now perfect.');
    decision = 2;
    result_status = 1;
else
    disp('BLAST still not working...')
    setenv('PATH', oldpath);
    pathmsg = strcat('PATH variable again updated to -> ', oldpath);
    disp(pathmsg);
    choice = questdlg('BLAST still not working. Press QUIT if you want to manually check your BLAST installation or press CONTINUE to let us install BLAST for you.', 'BLAST installation 1', 'QUIT', 'CONTINUE', 'CONTINUE');
    switch choice
    case 'QUIT'
        decision = 0;
    case 'CONTINUE'
        decision = 1;
    end
else
    error('Error:01 - Function aborted by the user.');
end
else
    disp('File blastinst.log not found in HOME directory.')
    disp('blastinst.log stores the path to the bin folder used by blast to run functions.')
    disp('Thus, with no blastinst.log file we can not automatically detect the database folder.')
    choice = questdlg('BLAST is not working properly. Press QUIT if you want to manually check your BLAST installation or press CONTINUE to let us install it for you.', 'BLAST installation 2', 'QUIT', 'CONTINUE', 'CONTINUE');
    switch choice
    case 'QUIT'
        decision = 0;
    case 'CONTINUE'
        decision = 1;
    end
end

% Install BLAST if decision = 1 (user choose CONTINUE)
if decision == 1
    % Let user choose in which folder BLAST should be installed.
    disp('Starting BLAST installation...')
    choice = questdlg('BLAST installation is an interactive process. You will be prompted to select folders and files required to continue the installation. Now, press OK and a new dialog box will let you choose a folder where BLAST will be installed.', 'BLAST installation 2', 'OK', 'OK');
    instfolder = uigetdir(matlabroot, 'Choose a folder to install BLAST')
cd(instfolder);

    % Let the user choose if he wants to get BLAST from the internet or
% from a local folder.
% Construct dialog box
choice = questdlg('Do you have BLAST installation files (e.g. ncbi-blast-2.2.26+-
universal-macosx.tar.gz) or do you want us to download it from the NCBI FTP server? The
installation files are compacted in a tar.gz file that MUST be downloaded from

'BLAST installation 3', 'Thanks. I have the files', 'Please, download it from NCBI', 'Please, download it from NCBI');

switch choice
    case 'Thanks. I have the files'
        decision2 = 0;
    case 'Please, download it from NCBI'
        decision2 = 1;
end

% Get the files from the place defined by the user
if decision2 == 0
    choice = questdlg('Now, you will be prompted to indicate where is the BLAST
installation file. Press ok and indicate the file you downloaded from NCBI.

'BLAST installation 4', 'OK', 'OK');
    [instfile, path] = uigetfile({'*.tar.gz'}, matlabroot);
    copyfile(fullfile(path,instfile), pwd());
else
    disp('Connecting to ftp.ncbi.nih.gov...');
    f = ftp('ftp.ncbi.nih.gov');
    cd(f,'blast/executables/LATEST');
    instfile = dir(f, '*macox.tar.gz');
    disp('Downloading the most recent BLAST version for MAC OS. It may take some
minutes...');
    mget(f, instfile.name);
    close(f);
    instfile = instfile.name;
    disp('BLAST executables successfully downloaded.');
end

% Initiate installation.
% Untar installation file
disp('Untaring installation file...');
untar(instfile);
disp('Untar done.');
% Update the $PATH environment variable to include the binfolder.
disp('Preparing PATH variable...');
blastfolder = regexp(instfile, '+', 'split');
blastfolder = strcat(blastfolder{1}, '+');
blastfolder = strcat(blastfolder, '/bin');
oldpath = getenv('PATH');
envpath = strcat(oldpath, ':', blastfolder);
disp('Setting PATH variable...');
setenv('PATH', char(envpath));
pathmsg = strcat('PATH variable updated from -> ', oldpath,' to -> ', cellstr(envpath));
disp(pathmsg);
% Create db folder, a folder to save databases, inside blastfolder.
disp('Creating database folder under BLAST installation folder...');
cd(char(blastfolder));
mkdir('db');
dbfolder = strcat(blastfolder, '/db');
disp('Database folder successfully created.');
disp(strcat('Creating .ncbirc file and saving it into ->',' dbfolder, '...'));
ncbircfile = ['[^BLAST]$'; strcat('.ncbirc', 'dbfolder)];
fid=fopen(strcat('.ncbirc file and saving it into ->',' dbfolder, '...'));
[rows,cols]=size(ncbircfile);
for p=1:rows
    fprintf(fid,'%s,','ncbircfile{p,1:end-1});
    fprintf(fid,'%s
','ncbircfile{p,end});
end
fclose(fid);
disp('.ncbirc file successfully created and saved.');
result_status = 1;
% Record results into a log file that will be saved into the % installation folder.
disp('Producing logfile...');
logfile = [instfolder; binfolder; dbfolder; oldpath; envpath; cellstr(date); instfile];
fid=fopen(strcat(getenv('HOME'), '/blastinst.log'), 'wt');
[rows, cols] = size(logfile);
for p=1:rows
    fprintf(fid, '%s,', logfile{p, end-1});
    fprintf(fid, '%s
', logfile{p, end});
end
fclose(fid);
disp(strcat('Logfile blastinst.log saved in ->', getenv('HOME')));
disp('Please, DO NOT delete this file. It will be used by M2N Toolbox in the future.');

if decision == 0
    error('Error:02 - Function aborted by the user.);
    result_status = 0
end

Appendix 6d.

MATLAB code for the function install_database() of M2N Toolbox.

function [result_status] = install_database()
% This function has been developed to install databases used for BLASTing.
% install_database installs databases preformated and available from the NCBI
% FTP server. For that, install_database uses the perl script
% update_blastdb.pl which is part of the BLAST installation files.
%
% Define a database folder and check which databases the user has available at the moment.
result_status = 0;
disp('Looking for the blastinst.log file...');
blastinstfile = exist(fullfile(char(getenv('HOME')), 'blastinst.log'), 'file');
if blastinstfile == 2
    disp('File found in the HOME directory.');
    disp('Loading blastinst.log...');
    blastinst = loadcell(fullfile(char(getenv('HOME')), 'blastinst.log'), ', ', '');
    disp('blastinst.log loaded');
    dbfolder = blastinst(3);
    choice = questdlg(strcat('The database folder is defined as ->', dbfolder, '. Press OK to confirm or press SELECT NEW to define a diferent database folder.'), 'Database installation', 'OK', 'SELECT NEW', 'OK');
    switch choice
        case 'OK'
            decision1 = 0;
        case 'SELECT NEW'
            decision1 = 1;
        end
    if decision1 == 1
        dbfolder = uigetdir(matlabroot, 'Choose the database folder.');
        disp(strcat('The new database folder is defined as ->', dbfolder));
    end
else
    disp('File blastinst.log not found in HOME directory.);
    disp('blastinst.log stores the path to the folder used by blast to save databases.');
    disp('Thus, with no blastinst.log file we can not automatically detect the database folder.');
choice = questdlg('As we could not detect the file blastinst.log, please press OK and a new dialog box will let you choose the folder you use to save databases.', 'Database installation 2', 'OK', 'OK');
dbfolder = uigetdir(matlabroot, 'Choose the database folder.');
end

% Ask user if he/she wants to install databases from NCBI server or from a local file.
choice = questdlg('Press CONTINUE to download databases from the NCBI server or press SELECT DB to select a local file containing the desired database. ATTENTION! Database files accepted here are compacted files in tar.gz format downloaded from ftp://ftp.ncbi.nlm.nih.gov/blast/db/.', 'Database installation 3', 'CONTINUE', 'SELECT DB', 'CONTINUE')
switch choice
    case 'CONTINUE'
        decision2 = 0;
    case 'SELECT DB'
        decision2 = 1;
end

% Inset databases according to the user decision, from NCBI server or from local file.
if decision2 == 0 % From NCBI server
    if blastinstfile == 2 % In case blastinstfile was found before
        binfolder = blastinst(2);
        databases = perl(fullfile(char(binfoolder), 'update_blastdb.pl'), '--showall');
        databases = regexp(databases, databases(18), 'split');
        databases = databases(2:(length(databases)-1));
        [datachosen, ok] = listdlg('ListString', databases);
        [download_data, status] = perl(fullfile(binfoolder, 'update_blastdb.pl'), char(databases(datachosen)));
    else
        choice = questdlg('We use a file called update_blastdb.pl to download databases from the NCBI server. However, as we could not detect the file blastinst.log, we are not sure where the file update_blastdb.pl could be located. There is a version of this file in the M2N Toolbox installation folder. Thus, press OK and a new dialog box will let you inform us in which folder M2N Toolbox has been installed.', 'Database installation 4', 'CONTINUE', 'SELECT DB', 'CONTINUE')
        instfolder = uigetdir(matlabroot, 'Choose the M2N Toolbox installation folder. ')
        updatefile = exist(fullfile(instfolder, 'update_blastdb.pl'))
        if updatefile == 2
            disp('Thank you, the file update_blastdb.pl has been found.')
            updatefile = fullfile(instfolder, 'update_blastdb.pl')
            databases = perl(updatefile, '--showall')
            databases = regexp(databases, databases(18), 'split')
            databases = databases(2:(length(databases)-1))
            [datachosen, ok] = listdlg('ListString', databases)
            [download_data, status] = perl(fullfile(updatefile, char(databases(datachosen))))
        else
            choice = questdlg('For some reason the file you chose is not the update_blastdb.pl. Could you please select it again? Press OK and a new dialog box will let you inform us where the file update_blastdb.pl is.', 'Database installation 5', 'CONTINUE', 'SELECT DB')
            [updatefile, updatepath] = uigetfile({'update_blastdb.pl'}, matlabroot)
            if strmatch(updatefile, 'update_blastdb.pl') == 1
                disp('Thank you. The file update_blastdb.pl has been found.')
                updatefile = fullfile(updatefile, updatepath)
                databases = perl(updatefile, '--showall')
                databases = regexp(databases, databases(18), 'split')
                databases = databases(2:(length(databases)-1))
                [datachosen, ok] = listdlg('ListString', databases)
                [download_data, status] = perl(fullfile(updatefile, updatepath), char(databases(datachosen)));
            else
                while strmatch(updatefile, 'update_blastdb.pl') ~= 1
                    choice = questdlg('For some reason the file you chose is not the update_blastdb.pl. Could you please select it again? Press OK and a new dialog box will let you inform us where the file update_blastdb.pl is.', 'Database installation 6', 'CONTINUE', 'SELECT DB')
                    [updatefile, updatepath] = uigetfile({'update_blastdb.pl'}, matlabroot)
                end
            end
        end
    end
end
Appendix 6e.

MATLAB code for the function databases() of M2N Toolbox.

```matlab
function [databases result_status] = databases()
% This function has been developed to verify BLAST databases installed locally.
% 
% result_status = 0;
% disp('Looking for the blastinst.log file...');
% blastinstfile = exist(fullfile(char(getenv('HOME')),'blastinst.log'), 'file');
if blastinstfile == 2
    disp('File found in the HOME directory.');
    disp('Loading blastinst.log...');
    blastinst = loadcell(fullfile(char(getenv('HOME')),'blastinst.log'), ',', '');
    disp('blastinst.log loaded');
    dbfolder = blastinst(3);
    choice = questdlg(strcat('The database folder is defined as ->', dbfolder, '. Press OK to confirm or press SELECT NEW to define a diferent database folder. '), 'Database installation', 'OK', 'SELECT NEW', 'OK');
    switch choice
        case 'OK'
            decision1 = 0;
        case 'SELECT NEW'
            decision1 = 1;
    end
if decision1 == 1
```

The rest of the code is similar to the provided text.
dbfolder = uigetdir(matlabroot, 'Choose the database folder.');
if dbfolder == 0
  error('Error 01 - databases: Function aborted by the user.);
else
  disp(strcat('The new database folder is defined as ->', dbfolder));
end

else
  disp('File blastinst.log not found in HOME directory.);
  disp('blastinst.log stores the path to the folder used by blast to save databases.');
  disp('Thus, with no blastinst.log file we can not automatically detect the database folder.');
  choice = questdlg('As we could not detect the file blastinst.log, please press OK and a new dialog box will let you choose the folder you use to save databases.', 'Database installation 2', 'OK', 'OK');
  dbfolder = uigetdir(matlabroot, 'Choose the database folder.');
end

% List databases already installed in the database folder
oldfolder = pwd();
cd(dbfolder);
databases = dir('*_pal*');
databases = struct2cell(databases);
databases = databases(1,:);
databases = regexpprep(databases, '.pal', '');
databases = transpose(databases);
cd(oldfolder);
result_status = 1;

Appendix 6f.

MATLAB code for the function MET2EC() of M2N Toolbox.

function [result_met] = MET2EC(data, header, folder, organism, save)
%This function was developed to assign EC numbers to metabolites.
%function [ecnumbers] = MET2EC(data, folder, organism);
% MET2EC makes use of the KEGG database to assign potential enzymes related to the metabolism of a specific organism.

% DATA is a character string naming the path to a csv file containing the input data in the following format: the first column contains the list of metabolites and the second column contains their respective KEGG codes (Figure 1 and supplementary file Example_input_data.csv). The default behavior of DATA is to open a dialog box where the user % ORGANISM = a character string containing the KEGG code for the organism under analysis (e.g. ?illa? for Lactococcus lactis IL1403). % The list of KEGG codes for organisms can be found at http://www.genome.jp/kegg/catalog/org_list.html.

% Collect arguments
% Collect data
keggcol = 2;
if exist('data', 'var')
  if ischar(data)
    datafile = exist(char(data), 'file');
    if datafile == 2
      disp(strcat('File <', data, '> found. Loading file...'));
      dataset = loadcell(data, ',', '');
    else
      disp('Input data file loaded.');
    end
  end
end

result_met = MET2EC(data, header, folder, organism, save);
choice = questdlg('Sorry, but we could not find the input file. Press OK and a new
dialog box will let you inform us where the input file is located. It must be in CSV format
and containing the name of compounds in the first column and their respective KEGG codes in
the second column.', 'MET2EC', 'OK', 'OK');
[dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
dataset = loadcell(fullfile(datapath, dataset), ',', ' ');
disp('Input data file loaded.');
end
else
dataset = data;
end else
choice = questdlg('Press OK and a new dialog box will let you inform us where the input
file is located. It must be in CSV format and containing the name of compounds in the first
column and their respective KEGG codes in the second column.', 'MET2EC 2', 'OK', 'OK');
[dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
dataset = loadcell(fullfile(datapath, dataset), ',', ' ');
disp('Input data file loaded.');
end
[row, column] = size(dataset);
if column ~= 2
  if column > 2
    datasethead = dataset(1,:);
    for i = 1:length(datasethead)
      if ~ischar(datasethead{i});
        datasethead(i) = [];
      end
    end
  end
  choice = questdlg('For some reason your input data has more columns than it was
supposed to have. The input data consists in a cell array of strings containing the name of
compounds in the first column and their respective KEGG codes in the second column.', 'MET2EC 3', 'OK', 'OK');
  [cpdcolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300],
    'SelectionMode', 'single', 'Name', 'Select compound names', 'PromptString', 'Select the column
containing names of compounds');
  if ok == 1
    cpdcolumn = datasethead{cpdcolumn};
    cpdcolumn = regexp(dataset(1,:), cpdcolumn, 'match');
    [nouse columncpd] = size(cpdcolumn);
    for i = 1:columncpd
      if ~isempty(cpdcolumn{i})
        thisone = i;
      end
    end
    if exist('thisone', 'var')
      compounds = dataset(:,thisone);
    else
      error('We could not fix the problem with the input data. Please check the
format of the input data and try again.');
    end
  else
    error('Error:01 - Function aborted by the user.');
  end
  choice = questdlg('Now, press OK to choose the column containing the KEGG CODES or
press CANCEL to abort the function.', 'MET2EC 4', 'OK', 'OK');
  [keggcolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300],
    'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column
containing KEGG codes');
  if ok == 1
    keggcolumn = datasethead{keggcolumn};
    keggcolumn = regexp(dataset(1,:), keggcolumn, 'match');
    [nouse columnkegg] = size(keggcolumn);
    for i = 1:columnkegg
      if ~isempty(keggcolumn{i})
        thisone = i;
      end
    end
    if exist('thisone', 'var')
      compounds = dataset(:,thisone);
    else
      error('We could not fix the problem with the input data. Please check the
format of the input data and try again.');
    end
  else
    error('Error:01 - Function aborted by the user.');
  end
  choice = questdlg('Now, press OK to choose the column containing the KEGG CODES or
press CANCEL to abort the function.', 'MET2EC 4', 'OK', 'OK');
  [keggcolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300],
    'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column
containing KEGG codes');
  if ok == 1
    keggcolumn = datasethead{keggcolumn};
    keggcolumn = regexp(dataset(1,:), keggcolumn, 'match');
    [nouse columnkegg] = size(keggcolumn);
    for i = 1:columnkegg
      if ~isempty(keggcolumn{i})
        thisone = i;
      end
    end
  end
otherone = i;

if exist('otherone', 'var')
    keggcodes = dataset(:,otherone);
else
    error('We could not fix the problem with the input data. Please check the format of the input data and try again.');
end

if column == 1
    choice = questdlg('Your input data has only one column. The correct input data should have 2 columns: one containing the name of compounds and another one containing their respective KEGG codes. We will assume the input data you used now is a list of KEGG codes. Press CONTINUE to try this way or press QUIT to abort the function.', 'MET2EC 4', 'CONTINUE', 'QUIT', 'CONTINUE');
    switch choice
    case 'CONTINUE'
        decision = 0;
    case 'QUIT'
        decision = 1;
    end
    if decision == 1
        error('Error:03 - Function aborted by the user.');
    end
    keggcol = 1;
end

% Check for duplicates
[r c] = size(dataset);
[a b total] = unique(dataset(:,c), 'first');
if length(b) ~= length(total)
    duplicated = dataset;
    duplicated(b,:) = [];
    disp(duplicated);
    choice = questdlg('The input data has some compounds sharing the same KEGG code. As we use KEGG codes to look for potential enzymes, duplicated KEGG codes will just duplicate the analysis time. The rows printed on screen have their KEGG codes defined for other compounds. Press CONTINUE to delete this rows and continue or press QUIT to abort the function.', 'MET2EC 5', 'CONTINUE', 'QUIT', 'CONTINUE');
    switch choice
    case 'CONTINUE'
        decision = 0;
    case 'QUIT'
        decision = 1;
    end
    if decision == 0
        dataset = dataset(sort(b), :);
    else
        error('Error:03 - Function aborted by the user.');
    end
end

% Check if the input data has a header
if exist('header', 'var')
    if ~islogical(header)
        choice = questdlg('We are not sure if your input data has a header or not. Press HEADER if your input data actually has a header or press NO HEADER.', 'MET2EC 8', 'HEADER', 'NO HEADER', 'HEADER');
        switch choice
        case 'HEADER'
            h = 2;
    end
end
% Collect folder
if exist('folder', 'var')
    if ischar(folder)
        folderpath = exist(char(folder), 'dir');
        if folderpath == 7
            disp('Folder successfully set.);
        else
            choice = questdlg('Sorry, but we could not find the folder you specified. Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
            folder = uigetdir(matlabroot, 'Folder - results');
            if folder == 0
                error('Error:04 - Function aborted by the user.');
            else
                disp('Folder successfully set.);)
        end
    else
        disp('FOLDER must be character string.);
        choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
        folder = uigetdir(matlabroot, 'Folder - results');
        if folder == 0
            error('Error:05 - Function aborted by the user.');
        else
            disp('Folder successfully set.);
        end
    else
        choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
        folder = uigetdir(matlabroot, 'Folder - results');
        if folder == 0
            error('Error:06 - Function aborted by the user.');
        else
            disp('Folder successfully set.);
        end
    end
end

cd(folder)

% Connect to KEGG and collect organism
disp('Checking if you have KEGG API tools installed in this working folder...');
iskegghere = exist('@KEGG', 'dir');
if iskegghere == 7
    disp('You already have KEGG API tools installed.');
else
    disp('NO HEADER')
    h = 1;
else
    if header == true
        h = 2;
    else
        h = 1;
    end
end
else
    choice = questdlg('Press HEADER if your input data has a header or press NO HEADER.', 'METZEC 3', 'HEADER', 'NO HEADER', 'HEADER');
    switch choice
        case 'HEADER'
            h = 2;
        case 'NO HEADER'
            h = 1;
        end
end

% Collect folder
if exist('folder', 'var')
    if ischar(folder)
        folderpath = exist(char(folder), 'dir');
        if folderpath == 7
            disp('Folder successfully set.');
        else
            choice = questdlg('Sorry, but we could not find the folder you specified. Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
            folder = uigetdir(matlabroot, 'Folder - results');
            if folder == 0
                error('Error:04 - Function aborted by the user.');
            else
                disp('Folder successfully set.');
            end
        end
    else
        disp('FOLDER must be character string.);
        choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
        folder = uigetdir(matlabroot, 'Folder - results');
        if folder == 0
            error('Error:05 - Function aborted by the user.');
        else
            disp('Folder successfully set.');
        end
    end
else
    choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'METZEC 3', 'OK', 'OK');
    folder = uigetdir(matlabroot, 'Folder - results');
    if folder == 0
        error('Error:06 - Function aborted by the user.');
    else
        disp('Folder successfully set.');
    end
end

cd(folder)
kegg = KEGG;
classType = class(kegg);
else
disp('Installing KEGG API tools...');
wsdlURL = 'http://soap.genome.jp/KEGG.wsdl';
className = createClassFromWsdl(wsdlURL);
kegg = KEGG;
classType = class(kegg);
end
disp('Checking if the organism is properly set.');

% Get list of organisms from KEGG
orglist = list_organisms(kegg);
orglist = struct2cell(orglist);
orgcodes = orglist(1,:);
orgnames = orglist(2,:);

if exist('organism', 'var') % if organism defined by user
    if ischar(organism) % If is char
        if ismember(organism, orgcodes) % If organism is part of the KEGG codes available in KEGG
            orgname = strmatch(organism, orgcodes);
            orgname = orgnames(orgname);
            disp(strcat('Organism set as <', orgname, '>.'));
        else % When the code specified by the user is not matching with the ones available in KEGG
            orgcode = strmatch(organism, orgnames);
            orgcode = orgcodes(orgcode);
            disp(strcat('The KEGG code for your organism is <', orgcode, '>.'));
            organism = orgcode;
        end
    else % If the full name of the organism was used instead
        orgcode = strmatch(organism, orgnames);
        orgcode = orgcodes(orgcode);
        disp(strcat('The KEGG code for your organism is <', orgcode, '>.'));
        organism = orgcode;
    end
end

if decision == 0 % if the user selected an option from the list
    switch choice
        case 'CONTINUE'
            decision = 0;
        case 'QUIT'
            decision = 1;
        end
end

if decision == 1 % if the function was aborted by the user
    error('Error:02 - Function aborted by the user.');
if
else
    disp('ORGANISM must be character string.');
    choice = questdlg('Press OK and choose an organism to be used in this analyse.', 'MET2EC 6', 'OK', 'OK');
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        disp(strcat('Organism set as <', organism, '>.'));
    end
end
end
else
    disp('Press OK and choose an organism to be used in this analyse.');
    choice = questdlg('Press OK and choose an organism to be used in this analyse.', 'MET2EC 6', 'OK', 'OK');
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        disp(strcat('Organism set as <', organism, '>.'));
    end
end
end
end
choice = questdlg('MET2EC requires an organism to confirm the association with potential enzymes. If your organism is not present in the list is because this organism is not part of KEGG database. Press CONTINUE to see the list of organisms again or press QUIT to exit MET2EC.', 'MET2EC 7', 'CONTINUE', 'QUIT', 'CONTINUE');
switch choice
    case 'CONTINUE'
        decision = 0;
    case 'QUIT'
        decision = 1;
end
if decision == 0
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        error('Error:04 - You have not selected a proper organism code. The function has been aborted.');
    end
end
end
else
    disp('Press OK and choose an organism to be used in this analyse.');
    choice = questdlg('Press OK and choose an organism to be used in this analyse.', 'MET2EC 6', 'OK', 'OK');
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        error('Error:04 - You have not selected a proper organism code. The function has been aborted.');
    end
end
end
end
has been aborted.';
    end
end
if decision == 1
    error('Error:05 - Function aborted by the user.');
end
end
if exist('save', 'var')
    if islogical(save)
        % Do nothing
    else
        choice = questdlg('We are not sure if you want to save the results into a CSV file or not. Please, press SAVE if you want the results to be saved in the folder you specified before or press No, THANKS to see the results simply printed on the screen', 'MET2EC 8', 'SAVE', 'NO, THANKS', 'SAVE');
        switch choice
            case 'SAVE'
                decision = 0;
            case 'No, THANKS'
                decision = 1;
            end
        if decision == 0
            save = true;
        else
            save = false;
        end
    end
else
    choice = questdlg('Please, press SAVE if you want the results to be saved in the folder you specified before or press No, THANKS to have the results simply printed on the screen', 'MET2EC 9', 'SAVE', 'NO, THANKS', 'SAVE');
    switch choice
        case 'SAVE'
            decision = 0;
        case 'No, THANKS'
            decision = 1;
        end
    if decision == 0
        save = true;
    else
        save = false;
    end
end

% Finish collecting arguments
% Get enzymes by compounds
[rows, columns] = size(dataset);
disp('Collecting enzymes from KEGG... Please wait."

% Progress bar
bar = waitbar(0, 'Collecting enzymes from KEGG...');
for i = h:length(dataset); % h is part of the check header.
    enzymes = get_enzymes_by_compound(kegg, dataset(i,columns));
    enzymes = cellstr(enzymes);
    if ~isempty(enzymes);
        dataset(i,columns+1) = cellstr(strcat(enzymes{:}));
    else
        dataset(i,columns+1) = cellstr('');
    end
    waitbar(i/rows, bar, 'Collecting enzymes from KEGG...');
end
close(bar);
% Save the results
if save == true
    fid=fopen('data_enzymes.csv','wt');
    [rows,cols]=size(dataset);
    for i=1:rows;
        fprintf(fid,'%s,' ,dataset{i,1:end-1});
        fprintf(fid,'%s
' ,dataset{i,end});
    end
    fclose(fid);
end

% Check which enzymes are associated to my organism in KEGG database
disp('Connecting to KEGG database to verify enzymes potentially associated with my organism...')

% For each compound: check if each enzyme related to it has already been defined for my organism in KEGG and save a file showing the results.
[rw, cn] = size(dataset);
for j = h:row;
    perrow = dataset(j,:);
    if isempty(perrow(cn))
        % Do nothing
    else
        enzymes = cellstr(perrow(cn));
        enzymes = regexp(enzymes, 'ec:','split');
        enzymes = enzymes{:};
        if isempty(enzymes{1})
            enzymes{1} = [];
        end
        [r c] = size(enzymes);
        add = repmat(perrow(1,columns), 1, c);
        result_m = [enzymes(:) , add(:)];
        if exist('result_met', 'var')
            result_met = [result_met ; result_m];
        else
            result_met = result_m;
        end
    end
end

% Finding genes
[r c] = size(result_met);

% Progress bar
bar = waitbar(0, 'Checking for association between enzymes and organism...');
for i = 1:r
    defined = get_genes_by_enzyme(kegg, strcat('ec:',result_met(i,1)), lower(organism));
    if isempty(defined)
        result_met(i,3) = cellstr('none');
    else
        if r2 > 1
            result_met(i,3) = cellstr(strcat(defined{:}));
        else
            result_met(i,3) = cellstr(defined);
        end
    end
    waitbar(i/r, bar, 'Checking for association between enzymes and organism...');
end
close(bar);

% Include compounds names
disp('Adding compound name..');
if keggcol == 2
    [r c] = size(result_met);
    for i = 1:r
        keggcode = result_met(i,2);
```matlab
datasetrow = strmatch(keggcode, dataset(:,2));
result_met(i,4) = cellstr(dataset(datasetrow, 1));

end

end

% Preparing results for saving
if save == true
    fid=fopen('met2ec.csv','wt');
    [rows,cols]=size(result_met);
    for i=1:rows;
        fprintf(fid,'%s,',result_met{i,1:end-1});
        fprintf(fid,'%s
',result_met{i,end});
    end
    fclose(fid);
end

%%%%%%%%%%%%%%%%%%%%%
%%%%%% THE END %%%%%%%%%%%%%%%%%%%%%

Appendix 6g.
MATLAB code for the function EC2BLAST() of M2N Toolbox.

function [full_report shortlist] = EC2BLAST(data, header, folder, save, pdb_EC, pdblib_fasta, blastLocal, database, entrez, evalue)
%This function was developed to perform BLAST comparisons on protein sequences of EC numbers collected using MET2EC.
%function [ecnumbers] = MET2EC(data, folder, organism);
% EC2BLAST makes use of the EBI database to define the protein sequences of enzymes collected using MET2EC. We use BLAST to compare these sequences against a protein database such as nr or refseq_protein. Options for modifying BLAST algorithm are available.
%
% DATA is a character string naming the path to a csv file containing
% the input data in the following format: the first column contains
% the list of metabolites and the second column contains their respective
% KEGG codes (Figure 1 and supplementary file Example_input_data.csv). The
% default behavior of DATA is to open a dialog box where the user
%
% ORGANISM = a character string containing the KEGG code for
% the organism under analysis (e.g. ?lla? for Lactococcus lactis IL1403).
% The list of KEGG codes for organisms can be found at

%%%%%%%%%%%%%%%%%%%%%%%%%%
% Collect data
if exist('data', 'var')
    if ischar(data)
        datafile = exist(char(data), 'file');
        if datafile == 2
            disp(strcat('File <', data, '> found. Loading file...'));
            dataset = loadcell(data, ',', ');
            disp('Input data file loaded.');
        else
            choice = questdlg('Sorry, but we could not find the input file. Press OK and a new
dialog box will let you inform us where the input file is located. It must have exactly the
same format as the CSV file produced by MET2EC (data_genes.csv), which has the following
columns: EC numbers, KEGG codes for the compounds, KEGG codes for related genes and compound
name.', 'EC2BLAST', 'OK', 'OK');
            [dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
            dataset = loadcell(fullfile(datapath, dataset), ',', ');
            disp('Input data file loaded.');
        end
    else
        dataset = data;
    end
```
else
    choice = questdlg('Press OK and a new dialog box will let you inform us where the input file is located. It must have exactly the same format as the CSV file produced by MET2EC (data_genes.csv), which has the following columns: EC numbers, KEGG codes for the compounds, KEGG codes for related genes and compound name.', 'EC2BLAST 2', 'OK', 'OK');
    [dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
    dataset = loadcell(fullfile(datapath, dataset), ',', ' ');
    disp('Input data file loaded.');
end
[row, column] = size(dataset);
if column == 4
    if column > 4
        datasethead = ['Column_1'; 'Column_2'; 'Column_3'; 'Column_4'];
    else
        datasethead = ['Column_1'; 'Column_2'; 'Column_3'];
    end
    choice = questdlg('For some reason your input data has more or less columns than it was supposed to have. The input data consists in a cell array of strings containing the EC number of enzymes in the first column, the KEGG code of compounds in the second column, the KEGG codes of genes in the third column and the name of each compound in the fourth column. The fourth column is not essential, but the others are. We will try to fix the problem with your help. Press OK to choose the column containing the EC numbers.', 'EC2BLAST 3', 'OK', 'OK');
    [eccolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select EC number', 'PromptString', 'Select the column containing EC numbers');
    if ok == 1
        ecnumbers = dataset(:,eccolumn);
    else
        error('Error:01 - Function aborted by the user.');
    end
    choice = questdlg('Now, press OK to choose the column containing the KEGG CODES of compounds.', 'EC2BLAST 4', 'OK', 'OK');
    [keggcpdcolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column containing KEGG codes for compounds');
    if ok == 1
        keggcpdcodes = dataset(:,keggcpdcolumn);
    else
        error('Error:02 - Function aborted by the user.');
    end
    choice = questdlg('Now, press OK to choose the column containing the KEGG CODES of genes.', 'EC2BLAST 5', 'OK', 'OK');
    [kegggenecolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column containing KEGG codes for genes');
    if ok == 1
        kegggenecodes = dataset(:,kegggenecolumn);
    else
        error('Error:02 - Function aborted by the user.');
    end
    if column > 4
        choice = questdlg('Now, press OK to choose the column containing the name of compounds. It is not an essential column. You may press SKIP if you do not have a column containing the name of compounds.', 'EC2BLAST 4', 'OK', 'OK');
        [namecolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select compound name', 'PromptString', 'Select the column containing compound names', 'OKString', 'SELECT', 'CancelString', 'SKIP');
        if ok == 1
            cpdname = dataset(:,namecolumn);
            dataset = [ecnumbers keggcpdcodes kegggenecodes cpdname];
        else
            dataset = [ecnumbers keggcpdcodes kegggenecodes];
        end
    else
        dataset = [ecnumbers keggcpdcodes kegggenecodes];
    end
end
% Check if the input data has a header
if exist('header', 'var')
    if ~islogical(header)
        choice = questdlg('We are not sure if your input data has a header or not. Press HEADER if your input data actually has a header or press NO HEADER.', 'EC2BLAST 5', 'HEADER', 'NO HEADER', 'HEADER');
        switch choice
            case 'HEADER'
                h = 2;
            case 'NO HEADER'
                h = 1;
        end
    else
        if header == true
            h = 2;
        else
            h = 1;
        end
    end
else
    choice = questdlg('Press HEADER if your input data has a header or press NO HEADER.', 'EC2BLAST 6', 'HEADER', 'NO HEADER', 'HEADER');
    switch choice
        case 'HEADER'
            h = 2;
        case 'NO HEADER'
            h = 1;
    end
end

% Collect folder
if exist('folder', 'var')
    if ischar(folder)
        folderpath = exist(char(folder), 'dir');
        if folderpath == 7
            disp('Folder successfully set.');
        else
            choice = questdlg('Sorry, but we could not find the folder you specified. Press OK and a new dialog box will let you inform us where the results are to be saved.', 'EC2BLAST 7', 'OK', 'OK');
            folder = uigetdir(matlabroot, 'Folder - results');
            if folder == 0
                error('Error:04 - Function aborted by the user.);
            else
                disp('Folder successfully set.');
        end
    else
        disp('FOLDER must be character string.);
        choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'EC2BLAST 8', 'OK', 'OK');
        folder = uigetdir(matlabroot, 'Folder - results');
        if folder == 0
            error('Error:05 - Function aborted by the user.);
        else
            disp('Folder successfully set.');
    end
else
    choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'EC2BLAST 9', 'OK', 'OK');
    folder = uigetdir(matlabroot, 'Folder - results');
    if folder == 0
        error('Error:06 - Function aborted by the user.);
    else
disp('Folder successfully set.');
end

cd(folder)

% Collect save
if exist('save', 'var')
    if islogical(save)
        % Do nothing
    else
        choice = questdlg('We are not sure if you want to save the results into a CSV file or not. Please, press SAVE if you want the results to be saved in the folder you specified before or press No, THANKS to see the results simply printed on the screen', 'EC2BLAST 10', 'SAVE', 'NO, THANKS', 'SAVE');
        switch choice
            case 'SAVE'
                decision = 0;
            case 'No, THANKS'
                decision = 1;
        end
        if decision == 0
            save = true;
        else
            save = false;
        end
    end
else
    choice = questdlg('Please, press SAVE if you want the results to be saved in the folder you specified before or press No, THANKS to have the results simply printed on the screen', 'EC2BLAST 11', 'SAVE', 'NO, THANKS', 'SAVE');
    switch choice
        case 'SAVE'
            decision = 0;
        case 'No, THANKS'
            decision = 1;
    end
    if decision == 0
        save = true;
    else
        save = false;
    end
end

% Collect pdb_EC
if exist('pdb_EC', 'var')
    if ischar(pdb_EC)
        pdb_ECfile = exist(char(pdb_EC), 'file');
        if pdb_ECfile == 2
            disp(strcat('File <', pdb_EC, ' > found. Loading file...'));
            pdb_ECfile = loadcell(pdb_EC, ' ', '');
            disp('pdb_EC file loaded.');
        else
            choice = questdlg('Sorry, we could not find the file pdb_EC. Press OK to inform us where the pdb_EC file is located or press INTERNET to collect the file from the EBI server (requires internet connection). This file maps EC numbers to protein IDs.', 'EC2BLAST 12', 'OK', 'INTERNET', 'OK');
            switch choice
                case 'OK'
                    decision = 1;
                case 'INTERNET'
                    decision = 0;
            end
            if decision == 1
                [pdb_ECfile, datapath] = uigetfile({'*.*'}, matlabroot);
                pdb_ECfile = loadcell(fullfile(datapath, pdb_ECfile), ' ', '');
            end
    end
end
disp('pdb_EC file loaded.');
else
disp('Downloading pdb_EC from the EBI server. It may take some minutes.')
urlwrite('http://www.ebi.ac.uk/thornton-srv/databases/pdbsum/data/pdb_EC', 'pdb_EC');
disp('File downloading...');
pdb_ECfile = loadcell('pdb_EC', '');
disp('File loaded...');
end
end
else
    pdb_ECfile = pdb_EC;
end
else
    choice = questdlg('Press OK to inform us where the pdb_EC file is located or press INTERNET to collect the file from the EBI server (requires internet connection). This file maps EC numbers to protein IDs.', 'EC2BLAST 13', 'OK', 'INTERNET', 'OK');
    switch choice
    case 'OK'
        decision = 1;
    case 'INTERNET'
        decision = 0;
    end
if decision == 1
    [pdb_ECfile, datapath] = uigetfile({'*.*'}, matlabroot);
disp('Loading file...');
pdb_ECfile = loadcell(fullfile(datapath, pdb_ECfile), '', '');
disp('pdb_EC file loaded.');
else
disp('Downloading pdb_EC from the EBI server. It may take some minutes.')
urlwrite('http://www.ebi.ac.uk/thornton-srv/databases/pdbsum/data/pdb_EC', 'pdb_EC');
disp('Downloading...');
pdb_ECfile = loadcell('pdb_EC', '', '');
disp('File loaded...');
end
end

% Collect pdblib_fasta
if exist('pdblib_fasta', 'var')
    if ischar(pdblib_fasta)
        pdblib_fastafile = exist(char(pdblib_fasta), 'file');
        if pdblib_fastafile == 2
            disp(strcat('File <', pdblib_fasta, '> found. Loading file...'));
pdblib_fastafile = loadcell(pdblib_fasta, '', '');
disp('pdblib_fasta file loaded.');
        else
            choice = questdlg('Sorry, we could not find the file pdblib_fasta. Press OK to inform us where the pdblib_fasta file is located or press INTERNET to collect the file from the EBI server (requires internet connection). This file maps protein IDs to amino acid sequences.', 'EC2BLAST 14', 'OK', 'INTERNET', 'OK');
            switch choice
            case 'OK'
                decision = 1;
            case 'INTERNET'
                decision = 0;
            end
if decision == 1
    [pdblib_fastafile, datapath] = uigetfile({'*.*'}, matlabroot);
disp('Loading file...');
pdblib_fastafile = loadcell(fullfile(datapath, pdblib_fastafile), '', '');
disp('pdblib_fasta file loaded.');
else
disp('Downloading pdblib_fasta file from the EBI server. It may take some minutes.')
urlwrite('http://www.ebi.ac.uk/thornton-srv/databases/pdbsum/data/pdblib.fasta', 'pdblib_fasta');
disp('Downloading...');
pdblib_fastafile = loadcell('pdblib_fasta', '', '');
end
end
disp('File loaded...');
end
end
else
    pdblib_fastafile = pdblib_fasta;
end
else
    choice = questdlg('Press OK to inform us where the pdblib_fasta file is located or press INTERNET to collect the file from the EBI server (requires internet connection). This file maps protein IDs to amino acid sequences.', 'EC2BLAST 15', 'OK', 'INTERNET', 'OK');
    switch choice
        case 'OK'
            decision = 1;
        case 'INTERNET'
            decision = 0;
    end
    if decision == 1
        [pdblib_fastafile, datapath] = uigetfile({'*.*'}, matlabroot);
        disp('Loading file...');
        pdblib_fastafile = loadcell(fullfile(datapath, pdblib_fastafile), ' ', '');
        disp('pdblib_fasta file loaded.');
    else
        disp('Downloading pdblib_fasta file from the EBI server. It may take some minutes.')
        urlwrite('http://www.ebi.ac.uk/thornton-srv/databases/pdbsum/data/pdblib.fasta', 'pdblib_fasta');
        disp('Downloading...');
        pdblib_fastafile = loadcell('pdblib_fasta', ' ', '');
        disp('File loaded...');
    end
end
%
% Check if user wants BLAST locally or through internet
if exist('blastLocal', 'var')
    if islogical(blastLocal)
        blast_decision = blastLocal
    else
        choice = questdlg('We are not sure which type of BLAST comparison you want to perform. Press BLAST LOCAL to use your local BLAST installation or press BLAST NCBI to perform BLAST searches through the internet (requires internet connection). Local BLAST is considerably faster than BLAST the NCBI server. If you prefer BLAST locally but do not have BLAST installed, press BLAST LOCAL and we check it for you.', 'EC2BLAST 1', 'BLAST LOCAL', 'BLAST NCBI', 'BLAST LOCAL');
        switch choice
            case 'BLAST LOCAL'
                blast_decision = 1;
            case 'BLAST NCBI'
                blast_decision = 0;
        end
    end
else
    choice = questdlg('Press BLAST LOCAL to use your local BLAST installation or press BLAST NCBI to perform BLAST searches through the internet (requires internet connection). Local BLAST is considerably faster than BLAST the NCBI server. If you prefer BLAST locally but do not have BLAST installed, press BLAST LOCAL and we check it for you.', 'EC2BLAST 1', 'BLAST LOCAL', 'BLAST NCBI', 'BLAST LOCAL');
    switch choice
        case 'BLAST LOCAL'
            blast_decision = 1;
        case 'BLAST NCBI'
            blast_decision = 0;
    end
end
%
% Check the database the user wants to use
if blast_decision == 0
    if exist('database', 'var')
        if ischar(database)
basetouse = database
else
  choice = questdlg('We could not detect the database you chose to BLAST against. Press SELECT to choose a database from a list or press SKIP to BLAST against nr database.', 'EC2BLAST 1', 'SELECT', 'SKIP', 'SELECT');
  switch choice
    case 'SELECT'
      database_decision = 1;
    case 'SKIP'
      database_decision = 0;
  end
  if database_decision == 1
    available_data = {'nr', 'refseq_protein', 'swissprot', 'pat', 'month', 'pdb', 'env_nr'}
    [database, ok] = listdlg('ListString', available_data, 'ListSize', [300 300], 'Name', 'Select database', 'SelectionMode', 'single', 'PromptString', 'Select the database to BLAST against.', 'OKString', 'SELECT', 'CancelString', 'CANCEL');
    if ok == 1
      basetouse = available_data(database)
    else
      error('Error 01 - Function aborted by the user.')
    end
  else
    basetouse = 'nr';
  end
else
  choice = questdlg('You have not defined a database to BLAST against. Press SELECT to choose a database from a list or press SKIP to BLAST against nr database.', 'EC2BLAST 1', 'SELECT', 'SKIP', 'SELECT');
  switch choice
    case 'SELECT'
      database_decision = 1;
    case 'SKIP'
      database_decision = 0;
  end
  if database_decision == 1
    available_data = {'nr', 'refseq_protein', 'swissprot', 'pat', 'month', 'pdb', 'env_nr'}
    [database, ok] = listdlg('ListString', available_data, 'ListSize', [300 300], 'Name', 'Select database', 'SelectionMode', 'single', 'PromptString', 'Select the database to BLAST against.', 'OKString', 'SELECT', 'CancelString', 'CANCEL');
    if ok == 1
      basetouse = available_data(database)
    else
      error('Error 02 - Function aborted by the user.')
    end
  else
    basetouse = 'nr';
  end
end
if blast_decision == 1
  % Check if BLAST is installed
  install_blast();
  % Check databases
  inst_database = databases();
  % Select databases
  [basetouse, ok] = listdlg('ListString', inst_database, 'ListSize', [500 300], 'Name', 'Select databases', 'PromptString', 'Select the databases to BLAST or press INSTALL to install a new database.', 'OKString', 'SELECT', 'CancelString', 'INSTALL');
  if ok == 0
    % Install databases
    install_database();
    inst_database = databases();
    [basetouse, ok] = listdlg('ListString', inst_database, 'ListSize', [500 300], 'Name', 'Select database', 'SelectionMode', 'multiple', 'PromptString', 'Select the databases to BLAST or press INSTALL to install a new database.', 'OKString', 'SELECT', 'CancelString', 'INSTALL');
  end
end
'Select databases', 'PromptString', 'Select the databases to BLAST or press QUIT to stop the function.', 'OKString', 'SELECT', 'CancelString', 'QUIT');

if ok == 1
    basetouse = inst_database(basetouse);
    [ro co] = size(basetouse);
    if ro > 1
        for i = 2:ro
            basetouse{i} = strcat({ ' '}, basetouse{i});
        end
    end
    basetouse = strcat(basetouse{:});
else
    error('Error 01 - Function aborted by the user.');
end
else
    basetouse = inst_database(basetouse);
    [ro co] = size(basetouse);
    if ro > 1
        for i = 2:ro
            basetouse{i} = strcat({ ' '}, basetouse{i});
        end
    end
    basetouse = strcat(basetouse{:});
end

% Check ENTREZ
if exist('entrez', 'var')
    if ischar(entrez)
        entrez_decision = 1
    else
        choice = questdlg('We could not detect a value for Entrez. This parameter is used to filter the comparisons performed by BLAST. For example, one may define here the species to BLAST against (e.g. Bacteria or escherichia_coli). Press OK to enter a value for entrez or press SKIP to BLAST against the whole database chosen. ATTENTION: the use of entrez requires internet connection.', 'EC2BLAST 1', 'OK', 'SKIP', 'OK');
        switch choice
            case 'OK'
                entrez_decision = 1;
            case 'SKIP'
                entrez_decision = 0;
        end
        if entrez_decision == 1
            entrez = inputdlg('Please, enter the evalue for entrez:','evalue',1,
                {'[Bacteria]'});
        else
            entrez = ''; 
        end
    end
else
    choice = questdlg('We could not detect a value for Entrez. This parameter is used to filter the comparisons performed by BLAST. For example, one may define here the species to BLAST against (e.g. Bacteria or escherichia_coli). Press OK to enter a value for entrez or press SKIP to BLAST against the whole database chosen. ATTENTION: the use of entrez requires internet connection.', 'EC2BLAST 1', 'OK', 'SKIP', 'OK');
    switch choice
        case 'OK'
            entrez_decision = 1;
        case 'SKIP'
            entrez_decision = 0;
    end
    if entrez_decision == 1
        entrez = inputdlg('Please, enter the evalue for entrez:','evalue',1,
            {'[Bacteria]'});
    else
        entrez = ''; 
    end
end
% Check evalue
if exist('evalue', 'var')
    if isnumeric(evalue)
        % Do nothing
    else
        choice = questdlg('No evalue entered. Press OK to define one or press SKIP to use the default value of 10^-10.', 'EC2BLAST 1', 'OK', 'SKIP', 'OK');
        switch choice
            case 'OK'
                evalue_decision = 1;
            case 'SKIP'
                evalue_decision = 0;
        end
        if evalue_decision == 1
            evalue = inputdlg('Please, enter the evalue for BLASTing:', 'evalue', 1, {'0.0000000001'});
            evalue = str2num(char(evalue));
        else
            evalue = 0.0000000001;
        end
    end
else
    choice = questdlg('No evalue entered. Press OK to define one or press SKIP to use the default value of 10^-10.', 'EC2BLAST 1', 'OK', 'SKIP', 'OK');
    switch choice
        case 'OK'
            evalue_decision = 1;
        case 'SKIP'
            evalue_decision = 0;
    end
    if evalue_decision == 1
        evalue = inputdlg('Please, enter the evalue for BLASTing:', 'evalue', 1, {'0.0000000001'});
        evalue = str2num(char(evalue));
    else
        evalue = 0.0000000001;
    end
end

% Prepare data for BLASTING
if h == 2
    dataset(1,:) = [];
end
[row col] = size(dataset);
nextcol = col + 1;
ec_numbers = length(dataset(:,1));
% Run analysis for each row of dataset. h is used in case the dataset has a % header.
bar = waitbar(0, 'Collecting protein ids from pdbEC...');
for i=1:ec_numbers
    % Keep going if there is a gene defined for that enzyme. If not, insert
    % NONE
    rowdata = dataset(i,:);
    if strmatch(rowdata{1,3}, 'none', 'exact')
        if exist('blast_rep', 'var')
            rowdata(1,nextcol) = cellstr('none');
            blast_rep = [blast_rep; rowdata];
        else
            blast_rep = rowdata;
            blast_rep(1,nextcol) = cellstr('none');
        end
    else
        rows_ebi = ind2sub(size(pdb_ECfile(:,2)), strmatch(rowdata{1,1}, pdb_ECfile(:,2), 'exact'));
        % Check if there is any protein with this EC number defined in EBI.
        if length(rows_ebi) == 0
            break; % Exit the loop if no protein is found in EBI.
        else
            % Collect protein IDs from EBI.
            protein_ids = pdb_ECfile(rows_ebi, :);
            % Add protein IDs to dataset.
            rowdata(1,nextcol) = cellstr(protein_ids{1,1});
            blast_rep = [blast_rep; rowdata];
        end
    end
end
% Display results
disp('Protein IDs collected from pdbEC:');
disp(blast_rep);
if exist('blast_rep', 'var')
    rowdata(1,nextcol) = cellstr('none');
    blast_rep = [blast_rep; rowdata];
else
    blast_rep = rowdata;
    blast_rep(1,nextcol) = cellstr('none');
end
else
    % In case of more than one protein, add rows to accommodate the results from EBI
    if length(rows_ebi) > 1
        rowdata2 = rowdata;
        for j = 2:length(rows_ebi)
            rowdata = [rowdata;rowdata2];
        end
        for k=1:length(rows_ebi);
            gene_symbol = pdb_ECfile{rows_ebi(k)};
            % Get the PDB ID.
            if length(gene_symbol) == 5
                gene_symbol = strcat('>', gene_symbol(1:4), ':', gene_symbol(5));
            else
                gene_symbol = strcat('>', gene_symbol(1:4), ':');
            end
            rowdata(k, nextcol) = cellstr(gene_symbol);
        end
        if exist('blast_rep', 'var')
            blast_rep = [blast_rep; rowdata];
        else
            blast_rep = rowdata;
        end
    else
        gene_symbol = pdb_ECfile{rows_ebi};
        % Get the PDB ID.
        if length(gene_symbol) == 5
            gene_symbol = strcat('>', gene_symbol(1:4), ':', gene_symbol(5));
        else
            gene_symbol = strcat('>', gene_symbol(1:4), ':');
        end
        rowdata(k, nextcol) = cellstr(gene_symbol);
        if exist('blast_rep', 'var')
            blast_rep = [blast_rep; rowdata];
        else
            blast_rep = rowdata;
        end
    end
end
waitbar(i/ec_numbers, bar, 'Collecting protein ids from pdbEC...');
end
close(bar)

% Collect protein sequences from pdblib
[r c] = size(blast_rep);
nextcol = c+1;
bar = waitbar(0, 'Collecting protein sequences from pdblib...');
for i = 1:r
    if strmatch(blast_rep{i,c}, 'none', 'exact')
        blast_rep{i,nextcol} = cellstr('none');
    else
        gene_symbol = blast_rep(i,c);
        %[conf location] = ismember(gene_symbol, pdblib_fastafile(:,1));
        protseq = find(strcmp(gene_symbol, pdblib_fastafile(:,1)));
        %if conf == 1
        %    protseq = pdblib_fastafile(location + 1);
        %else
        %    blast_rep{i, nextcol} = cellstr('none');
        %end
        blast_rep(i, nextcol) = protseq;
end
if ~isempty(protseq)
    protseq = pdblib_fastafile(protseq + 1);
    blast_rep(i, nextcol) = protseq;
else
    blast_rep(i, nextcol) = cellstr('none');
end

waitbar(i/r, bar, 'Collecting protein sequences from pdblib...');
end
close(bar)

% Blast each sequence collected before from EBI
[r c] = size(blast_rep);
nextcol = c+1;
bar = waitbar(0, 'BLASTing...');
for i = 1:r
    if strmatch(blast_rep{i,c}, 'none', 'exact')
        blast_rep(i,nextcol) = cellstr('none');
    else
        if blast_decision == 0
            protseq = blast_rep(i,c)
            id_blast = 0;
            while id_blast == 0
                try
                    [id_blast, RTOE] = blastncbi(char(protseq), 'blastp', 'Expect', evalue,'Entrez', entrez, 'Database', basetouse);
                    catch Warning
disp('Accessing NCBI...');
pause(20)
                end
            end
            while exist('blast_report') == 0;
                try
                    blast_report = getblast(id_blast, 'WaitTime', 0.02);
                    catch Warning
disp('Trying to download BLAST report from NCBI...');
pause(5)
                end
            end
            if isstruct(blast_report);
                hits = length(blast_report.Hits);
                %%% Put the results in the report
                for w=1:hits
                    length_protein = blast_report.Hits{1,w}.Length;
                    expected_hit = blast_report.Hits{1,w}.HSPs.Expect;
                    score = blast_report.Hits{1,w}.HSPs.Score;
                    name_hit = blast_report.Hits{1,w}.Name;
                    blast_rep(i,c+w) = cellstr(strcat('Length:',num2str(length_protein), '|E-value:', num2str(expected_hit), '|Score:', num2str(score), '|Name:', name_hit));
                end
            else
                blast_rep(i,nextcol) = cellstr('none');
            end
        else
            % Get the sequence to BLAST
            protseq = blast_rep(i,c);
            % Prepare the FASTA file
            if exist('protseq.fa', 'file')
                delete('protseq.fa')
            end
            if exist('blasthits.csv', 'file')
                delete('blasthits.csv')
            end
            fastawrite('protseq.fa', char(protseq));
            % Set up the parameters to BLAST
            param1 = 'blastp-query protseq.fa -db '; %+ basetouse
            param2 = ' -evalue ';
```matlab
evalue = num2str(evalue);
param3 = '-entrez_query'; % + entrez
entrez = entrez;
param4 = '-remote';
param5 = '-out blasthits.csv -outfmt "10 evalue bitscore sgi"';
blastsearch = strcat(param1, ' ', basetouse, param2, ' ', evalue, param3, ' ', entrez, param4, param5);
%blastsearch = [param1, basetouse, param2, evalue, param3, entrez, param5, param4];
blastsearch = system(char(blastsearch));
if blastsearch ~= 0
    blast_rep(i,nextcol) = cellstr('blast_error');
else
    blasthitsFile = dir('blasthits.csv')
    if blasthitsFile.bytes == 0
        blast_rep(i,nextcol) = cellstr('blast_error');
    else
        report = loadcell('blasthits.csv', ',', ' ');
        [row col] = size(report);
        for k = 1:row
            taxid = report(k, 3);
            param11 = 'blastdbcmd -entry ';
            param22 = '-db '; % + basetouse
            param33 = '-dbtype prot -out taxid.csv -outfmt %t'
            taxcollect = [param11, num2str(taxid{1}), param22, basetouse, param33];
            taxcollect = system(taxcollect);
            if taxcollect ~= 0
                blast_rep(i,c+k) = cellstr(strcat(num2str(report{k,1}), '-', num2str(report{k,2})), '-', num2str(report{k,3}));
            else
                taxid = loadcell('taxid.csv', ',', ' ');
                taxid = strcat(taxid{1,:});
                blast_rep(i,c+k) = cellstr(strcat(num2str(report{k,1}), '-', num2str(report{k,2})), '-', num2str(report{k,3}), '-', taxid));
            end
        end
    end
end
if exist('protseq.fa', 'file')
    delete('protseq.fa')
end
if exist('blasthits.csv', 'file')
delete('blasthits.csv')
end
end
waitbar(i/r, bar, 'BLASTing...');
full_report = blast_rep
%%% Debuging
fid=fopen('full_report.csv', 'wt');
[rows,cols]=size(blast_rep);
for p=1:rows
    fprintf(fid, '%s,', blast_rep{p,1:end-1});
    fprintf(fid, '%s
', blast_rep{p,end});
end
fclose(fid);
fclose('all');
blastNotFound = find(cellfun(@(x) strcmp(x, 'none'), blast_rep(:,7)));
blastError = find(cellfun(@(x) strcmp(x, 'blast_error'), blast_rep(:,7)));
keggNotFound = find(cellfun(@(x) strcmp(x, 'none'), blast_rep(:,3)));
delBlast = [blastNotFound; blastError];
toDelete = intersect(delBlast, keggNotFound);
```
blast_rep(toDelete,:) = [];
shortlist = blast_rep;

fid=fopen('shortlist.csv','wt');
[rows,cols]=size(shortlist);
for p=1:rows
    fprintf(fid,'%s,',shortlist(p,1:end-1));
    fprintf(fid,'%s
',shortlist(p,end));
end
fclose(fid);
close('all');

%%%%%%%%%%%%%%%%%%%%%%%%%%% THE END %%%%%%%%%%%%%%%%%%%%%%%%%%

Appendix 6h.

MATLAB code for the function BLAST2MAPS() of M2N Toolbox.

function [] = BLAST2MAPS(data, header, organism, folder, text_color, bg_color)
%This function was developed to perform BLAST comparisons on protein sequences of EC numbers
%collected using MET2EC.
%function [ecnumbers] = MET2EC(data, folder, organism);
% EC2BLAST makes use of the EBI database to define the protein sequences of enzymes
collected using MET2EC. We use BLAST to compare these sequences against a protein database
such as nr or refseq_protein. Options for modifying BLAST algorithm are available.
%
% DATA is a character string naming the path to a csv file containing
% the input data in the following format: the first column contains
% the list of metabolites and the second column contains their respective
% KEGG codes (Figure 1 and supplementary file Example_input_data.csv). The
% default behavior of DATA is to open a dialog box where the user
%
% ORGANISM = a character string containing the KEGG code for
% the organism under analysis (e.g. ?lla? for Lactococcus lactis IL1403).
% The list of KEGG codes for organisms can be found at
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Collect data
if exist('data', 'var')
    if ischar(data)
        datafile = exist(char(data), 'file');
        if datafile == 2
            disp(strcat('File <', data, '> found. Loading file...'));
            dataset = loadcell(data, ',',' ');;
            disp('Input data file loaded.');
        else
            choice = questdlg('Sorry, but we could not find the input file. Press OK and a new
dialog box will let you inform us where the input file is located. It must have exactly the
same format as the CSV file produced by EC2BLAST (shortlist.csv), which has at least the
following columns: EC numbers, KEGG codes for the compounds, KEGG codes for related genes and
compound name. ', 'BLAST2MAPS', 'OK', 'OK');
            [dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
            dataset = loadcell(fullfile(datapath, dataset), ',',' ');;
            disp('Input data file loaded.');
        end
    else
        dataset = data;
    end
end
else
    choice = questdlg('Press OK and a new dialog box will let you inform us where the input file is located. It must have exactly the same format as the CSV file produced by EC2BLAST (shortlist.csv), which has at least the following columns: EC numbers, KEGG codes for the compounds, KEGG codes for related genes and compound name.', 'BLAST2MAPS 2', 'OK', 'OK');
    [dataset, datapath] = uigetfile({'*.csv'}, matlabroot);
    disp('Loading file...please wait');
    dataset = loadcell(fullfile(datapath, dataset), ',', '');
    disp('Input data file loaded. ');
end
[row, column] = size(dataset);
if column < 4
    if column < 4
        datasethead = ['Column_1'; 'Column_2'; 'Column_3'; 'Column_4'; 'Column_5'; 'Column_6'; 'Column_7'];
    else
        datasethead = ['Column_1'; 'Column_2'; 'Column_3'; 'Column_4'];
    end
    choice = questdlg('For some reason your input data has less columns than it was supposed to have. The input data consists in a cell array of strings containing the EC number of enzymes in the first column, the KEGG code of compounds in the second column, the KEGG codes of genes in the third column and the name of each compound in the fourth column. The fourth column is not essential, but the others are. We will try to fix the problem with your help. Press OK to choose the column containing the EC numbers.', 'BLAST2MAPS 3', 'OK', 'OK');
    [eccolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select EC number', 'PromptString', 'Select the column containing EC numbers');
    if ok == 1
        ecnumbers = dataset(:,eccolumn);
    else
        error('Error:01 - Function aborted by the user.');
    end
    choice = questdlg('Now, press OK to choose the column containing the KEGG CODES of compounds.', 'EC2BLAST 5', 'OK', 'OK');
    [keggcpdcolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column containing KEGG codes for compounds');
    if ok == 1
        keggcpdcodes = dataset(:,keggcpdcolumn);
    else
        error('Error:02 - Function aborted by the user.');
    end
    choice = questdlg('Now, press OK to choose the column containing the KEGG CODES of genes.', 'EC2BLAST 5', 'OK', 'OK');
    [kegggenecolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select KEGG codes', 'PromptString', 'Select the column containing KEGG codes for genes');
    if ok == 1
        kegggenecodes = dataset(:,kegggenecolumn);
    else
        error('Error:02 - Function aborted by the user.');
    end
    choice = questdlg('Now, press OK to choose the column containing the name of compounds. It is not an essential column. You may press SKIP if you do not have a column containing the name of compounds.', 'BLAST2MAPS 4', 'OK', 'OK');
    [namecolumn, ok] = listdlg('ListString', datasethead, 'ListSize', [300 300], 'SelectionMode', 'single', 'Name', 'Select compound name', 'PromptString', 'Select the column containing compound names');
    if ok == 1
        cpdname = dataset(:,namecolumn);
        dataset = [ecnumbers keggcpdcodes kegggenecodes cpdname];
    else
        error('Error:02 - Function aborted by the user.');
    end
end
% Check if the input data has a header
if exist('header', 'var')
if ~islogical(header)
choice = questdlg('We are not sure if your input data has a header or not. Press
HEADER if your input data actually has a header or press NO HEADER.', 'BLAST2MAPS 5',
'HEADER', 'NO HEADER', 'HEADER');
switch choice
case 'HEADER'
h = 2;
case 'NO HEADER'
h = 1;
end
else
if header == true
h = 2;
else
h = 1;
end
end
else
choice = questdlg('Press HEADER if your input data has a header or press NO HEADER.',
'BLAST2MAPS 6', 'HEADER', 'NO HEADER', 'HEADER');
switch choice
case 'HEADER'
h = 2;
case 'NO HEADER'
h = 1;
end
end
% Connect to KEGG and collect organism
disp('Checking if you have KEGG API tools installed in this working folder...');
iskegghere = exist('@KEGG', 'dir');
if iskegghere == 7
disp('You already have KEGG API tools installed.');
kegg = KEGG;
classType = class(kegg);
else
disp('Installing KEGG API tools...');
wsdlURL = 'http://soap.genome.jp/KEGG.wsdl';
className = createClassFromWsdl(wsdlURL);
kegg = KEGG;
classType = class(kegg);
end
disp('Checking if the organism is properly set.');
% Get list of organisms from KEGG
orglist = list_organisms(kegg);
orglist = struct2cell(orglist);
orgcodes = orglist(1,:);
orgnames = orglist(2,:);
if exist('organism', 'var') % if organism defined by user
if ischar(organism) % If is char
if ismember(organism, orgcodes) % If organism is part of the KEGG codes available in
KEGG
orgname = strmatch(organism, orgcodes);
orgname = orgnames(orgname);
disp(strcat('Organism set as <', orgname,'>.'));
else
if ismember(organism, orgnames) % If the full name of the organism was used
instead
orgcode = strmatch(organism, orgnames);
orgcode = orgcodes(orgcode);
disp(strcat('The KEGG code for your organism is <', orgcode,'>.'));
organism = orgcode;
else % When the code specified by the user is not matching with the ones available

356


in KEGG

choice = questdlg('Sorry, but we could not recognise your organism. Press OK and choose the correct organism.', 'MET2EC 4', 'OK', 'OK');
[orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
if ok == 1
    organism = orgcodes(orgchosen);
    disp(strcat('Organism set as <', organism, '>.'));
else
    choice = questdlg('MET2EC requires an organism to confirm the association with potential enzymes. If your organism is not present in the list is because this organism is not part of KEGG database. Press CONTINUE to see the list of organisms again or press QUIT to exit MET2EC.', 'MET2EC 5', 'CONTINUE', 'QUIT', 'CONTINUE');
    switch choice
    case 'CONTINUE'
        decision = 0;
    case 'QUIT'
        decision = 1;
    end
    if decision == 0
        [orgchosen, ok] = listdlg('ListString', orgnames);
        if ok == 1
            organism = orgcodes(orgchosen);
            disp(strcat('Organism set as <', organism, '>.'));
        else
            error('Error:01 - You have not selected a proper organism code. The function has been aborted.');
        end
    end
    if decision == 1
        error('Error:02 - Function aborted by the user.');
    end
end
else
    disp('ORGANISM must be character string.');
    choice = questdlg('Press OK and choose an organism to be used in this analyse.', 'MET2EC 6', 'OK', 'OK');
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        choice = questdlg('MET2EC requires an organism to confirm the association with potential enzymes. If your organism is not present in the list is because this organism is not part of KEGG database. Press CONTINUE to see the list of organisms again or press QUIT to exit MET2EC.', 'MET2EC 7', 'CONTINUE', 'QUIT', 'CONTINUE');
        switch choice
        case 'CONTINUE'
            decision = 0;
        case 'QUIT'
            decision = 1;
        end
        if decision == 0
            [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
            if ok == 1
                organism = orgcodes(orgchosen);
                disp(strcat('Organism set as <', organism, '>.'));
            else
                error('Error:04 - You have not selected a proper organism code. The function has been aborted.');
            end
        end
    end
end
end
end
end
end
if decision == 1
    error('Error:05 - Function aborted by the user.');
end
end
else
    choice = questdlg('Press OK and choose an organism to be used in this analyse.', 'MET2EC 6', 'OK', 'OK');
    [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
    if ok == 1
        organism = orgcodes(orgchosen);
        disp(strcat('Organism set as <', organism, '>.'));
    else
        choice = questdlg('MET2EC requires an organism to confirm the association with potential enzymes. If your organism is not present in the list is because this organism is not part of KEGG database. Press CONTINUE to see the list of organisms again or press QUIT to exit MET2EC.', 'MET2EC 7', 'CONTINUE', 'QUIT', 'CONTINUE');
        switch choice
            case 'CONTINUE'
                decision = 0;
            case 'QUIT'
                decision = 1;
        end
    end
    if decision == 0
        [orgchosen, ok] = listdlg('ListString', orgnames, 'ListSize', [400 300], 'SelectionMode', 'single', 'Name', 'Select organism', 'PromptString', 'Select the organism you are studying');
        if ok == 1
            organism = orgcodes(orgchosen);
            disp(strcat('Organism set as <', organism, '>.'));
        else
            error('Error:04 - You have not selected a proper organism code. The function has been aborted.');
        end
    end
    if decision == 1
        error('Error:05 - Function aborted by the user.');
    end
end
if exist('bg_color', 'var');
    bg_color = {bg_color};
else
disp('Background color set as yellow. For changing colors use the argument bg_color');
    bg_color = {'#ffff00'};
end
if exist('text_color', 'var');
    text_color = {text_color};
else
disp('Text color set as red. For changing colors use the argument text_color');
    text_color = {'#ff0000'};
end
% Collect folder
if exist('folder', 'var')
    if ischar(folder)
        folderpath = exist(char(folder), 'dir');
        if folderpath == 7
            disp('Folder successfully set.');
        else
            choice = questdlg('Sorry, but we could not find the folder you specified. Press OK and a new dialog box will let you inform us where the results are to be saved.', 'BLAST2MAPS');
        end
    end
end
folder = uigetdir(matlabroot, 'Folder - results');
if folder == 0
    error('Error:04 - Function aborted by the user.');
else
    disp('Folder successfully set.');
end
else
disp('FOLDER must be character string.');
choice = questdlg('Press OK and a new dialog box will let you inform us where the results are to be saved.', 'BLAST2MAPS 8', 'OK', 'OK');
folder = uigetdir(matlabroot, 'Folder - results');
if folder == 0
    error('Error:05 - Function aborted by the user.');
else
    disp('Folder successfully set.');
end
end
cd(folder)

shortlist = dataset;

%% Get the list of metabolites present in the shortlist
metabolites = unique(shortlist(:,2));

%% Find the rows of shortlist containing info about each metabolite and shortlist the enzymes related to that.
x = shortlist(:,2); % I created this variable to search (in the code below) the rows showing each enzyme.

map_pathway = [cellstr('Compound'), cellstr('Enzyme'), cellstr('Pathway')];

cpm = 1; % Parameter to control if there is any pathway related to this compound
enz = 1; % Parameter to control if there is any pathway related to this enzyme

l = length(metabolites);
bar = waitbar(0,'Collecting pathways...');
for i = 1:length(metabolites)
    rows_shortlist = [cellfun(@(x) strcmp(x, metabolites{i}), x)];
    rows_shortlist = shortlist(rows_shortlist,:);
    enzymes = unique(rows_shortlist(:,1));
    % For each metabolite and enzyme, collect the pathways from KEGG database.
    paths_comp = get_pathways_by_compounds(kegg, strcat('cpd:',metabolites{i}));
    if ~isempty(paths_comp)
        paths_comp = regexprep(paths_comp, 'path:ko', '');
        paths_comp = regexprep(paths_comp, 'path:map', '');
        paths_comp = unique(paths_comp);
    else
        paths_comp = strcat(metabolites{i}, '-No pathway in Kegg');
        cpm = 0;
    end
    % For each enzyme related to this compound, search the pathways and compare with the pathways we found for this metabolite. Similar
360

```matlab
%% pathways will be added to map_pathway.
for e = 1:length(enzymes)
    paths_enzyme = get_pathways_by_enzymes(kegg, strcat('ec:', enzymes(e)));
    if ~isempty(paths_enzyme)
        paths_enzyme = regexprep(paths_enzyme, 'path:ko', '');
        paths_enzyme = regexprep(paths_enzyme, 'path:map', ''
        paths_enzyme = unique(paths_enzyme);
    else
        paths_enzyme = strcat(enzymes(e), '-No pathway in Kegg');
        enz = 0;
    end
    if cpm == 1 & enz == 1
        common = intersect(paths_enzyme, paths_comp); % Common pathways between metabolites and enzymes
        if ~isempty(common)
            for p = 1:length(common)
                add_common = [cellstr(metabolites(i)), cellstr(enzymes(e)),
                              cellstr(common(p))];
                map_pathway = [map_pathway; add_common];
            end
        end
    end
    if cpm == 1 & enz == 0
        for p = 1:length(paths_comp)
            add_common = [cellstr(metabolites(i)), cellstr('0'),
                          cellstr(paths_comp(p))];
            map_pathway = [map_pathway; add_common];
        end
    end
    if cpm == 0 & enz == 1
        for p = 1:length(paths_enzyme)
            add_common = [cellstr(metabolites(i)), cellstr(paths_enzyme(p)),
                          cellstr('Path_related_to_enzyme_only')];
            map_pathway = [map_pathway; add_common];
        end
    end
    cpm = 1; % Parameter to control if there is any pathway related to this compound
    enz = 1; % Parameter to control if there is any pathway related to this enzyme
    common = '';
    waitbar(i/l, bar, 'Collecting pathways...');
end
close(bar)
```

```matlab
x = map_pathway(:, 3);
y = [cellfun(@(x) strcmp(x, 'Path_related_to_enzyme_only'), x)];
map_pathway(y,:) = [];
```

```matlab
x = map_pathway(:, 3);
y = [cellfun(@(x) length(x), x)];
map_pathway(y<5,:) = [];
```

```matlab
[rows, cols] = size(map_pathway);
paths_unique = unique(map_pathway(2:rows,3)); % Get the id of pathways with no repetition
paths = map_pathway(2:(rows),3);
paths = cell2mat(paths); % Convert the list of pathways in a matrix
counting = histc(str2num(paths),unique(str2num(paths)));
% Count how many times each pathway appear
paths_unique = [paths_unique, num2cell(counting)]; % Put together the unique pathways and the number of occurrences
paths_unique = sortrows(paths_unique, -2); % Sort to have the ranking of pathways
```

```matlab
mkdir('pathway_maps');
```
path_list = list_pathways(kegg, lower(organism));
path_list = struct2cell(path_list);
path_id = regexprep(path_list{1,1}, strcat('path:', lower(organism)), '');

l = length(paths_unique);
bar = waitbar(0, 'Generating MAPS...');
for i = 1:l
    path_list = list_pathways(kegg, lower(organism));
    path_list = struct2cell(path_list);
    path_id = regexprep(path_list{1,1}, strcat('path:', lower(organism)), '');
    x = map_pathway(:,3);
    row_objects = [cellfun(@(x) strcmp(x, pathway), x)];
    row_objects = map_pathway(row_objects,:);
    object_list = [unique(row_objects{:,1}); unique(row_objects{:,2})];
    color1 = repmat(text_color, 1, length(object_list));
    color2 = repmat(bg_color, 1, length(object_list));
    map = color_pathway_by_objects(kegg, strcat('path:', lower(organism), pathway),
                                    object_list, color1, color2);
    [filestr, status] = urlwrite(map, char(strcat('pathway_maps/',
                                                      num2str(paths_unique{i,2}),
                                                      '-hits-', path_name(2), '.png')));
    disp(char(strcat('pathway_maps/',
                     num2str(paths_unique{i,2}),
                     '-hits-', path_name(2), '.png DONE!')));
end
waitbar(i/l, bar, 'Generating MAPS...');
end
close(bar)

%%%%%%%%%%%%%%%%%%%%%%%%%%% THE END %%%%%%%%%%%%%%%%%%%%%%%%%

Appendix 6i.

MATLAB code for the function BLAST2NET() of M2N Toolbox.

function [network] = BLAST2NET()
% This function was developed to build a metabolic network using the
% results produced by EC2BLAST.
%
% Collect arguments

% Collect data
if exist('data', 'var')
    if ischar(data)
        datafile = exist(char(data), 'file');
        if datafile == 2
            disp(strcat('File <', data, '> found. Loading file...'));
            dataset = loadcell(data, ',','
            disp('Input data file loaded.');
        else
            choice = questdlg('Sorry, but we could not find the input file. Press OK and a new
dialog box will let you inform us where the input file is located. It must be in CSV format
and containing the name of compounds in the first column and their respective KEGG codes in
the second column.','.MET2EC', 'OK', 'OK');
            [dataset, datapath] = uigetfile({'.csv'}, matlabroot);
            dataset = loadcell(fullfile(datapath, dataset), ',', '');
            disp('Input data file loaded.');
        end
    else
        disp(strcat('Error: data must be a string. Please run the script again with the correct'));
    end
else
    disp('Error: data not found. Please run the script again with the correct data.')
end
361
dataset = data;
end
else
    choice = questdlg('Press OK and a new dialog box will let you inform us where the input file is located. It must be in CSV format and containing the name of compounds in the first column and their respective KEGG codes in the second column.', 'MET2EC 2', 'OK', 'OK');
    [dataset, datapath] = uigetfile(['*.csv'], matlabroot);
    dataset = loadcell(fullfile(datapath, dataset), 'r', '');
    disp('Input data file loaded.'));
end
 [~, column] = size(dataset);
 if column == 1
    error('Error:01 - The input data seems to have only one column. However, the input data must have at least 2 columns: the first one containing the EC number of predicted enzymes and the second one containing the KEGG codes of their respective compounds.');
end
% Delete rows containing no compounds
toDelete = regexp(dataset(:,2), 'C', 'start');
toDelete = find(cellfun(@(x) length(x) ~= 1, toDelete));
if ~isempty(toDelete)
    dataset(toDelete,:) = [];
end
% Check for duplicates
duplicated = strcat(dataset(:,1), dataset(:,2));
[~, c] = size(duplicated);
[~, b total] = unique(duplicated(:,1), 'first');
if length(b) ~= length(total)
duplicated2 = duplicated;
duplicated2(b,:) = [];
dataset2 = dataset(sort(b), :);
end
compounds = dataset2(:,2);
enzymes = dataset2(:,1);

[~, b total] = unique(compounds, 'first');
if length(b) ~= length(total)
    compounds2 = compounds;
    compounds2(b,:) = [];
    disp(compounds2);
    compounds = compounds(sort(b), :);
end
[~, b total] = unique(enzymes, 'first');
if length(b) ~= length(total)
    enzymes2 = enzymes;
    enzymes2(b,:) = [];
    disp(enzymes2);
    enzymes = enzymes(sort(b), :);
end
% Build the matrix
components = length(compounds)+length(enzymes);
for i = 1:length(compounds)
    cpd = compounds(i);
    if exist('compmat', 'var')
        compmat = [compmat;repmat(0,1,components)];
    else
        compmat = [repmat(0,1,components)];
    end
    connect = ind2sub(size(dataset2(:,2)), strmatch(cpd, dataset2(:,2), 'exact'));
    enz = dataset2(connect, 1);
if length(enz) > 1
    for j = 1:length(enz)
        enz2 = ind2sub(size(enzymes(:)), strmatch(enz(j), enzymes(:), 'exact'));
        compmat(i,(length(compounds)+enz2)) = 1;
    end
else
    enz2 = ind2sub(size(enzymes(:)), strmatch(enz, enzymes(:), 'exact'));
    compmat(i,(length(compounds)+enz2)) = 1;
end
end

for i = 1:length(enzymes)
    enz = enzymes(i)
    if exist('compmat', 'var')
        compmat = [compmat;repmat(0,1,components)];
    else
        compmat = [repmat(0,1,components)];
    end

    % connect = ind2sub(size(dataset(:,1)), strmatch(enz, dataset(:,1), 'exact'))
    % cpd = dataset(connect, 2)
    % cpd = ind2sub(size(compounds(:)), strmatch(cpd, compounds(:), 'exact'))
    % compmat((length(compounds)+i),(length(enzymes)+cpd)) = 1
end
%clear('compmat')

compNames = dataset(:,[2,4])
for i = 1:length(compounds)
    NameRow = find(cellfun(@(x) strcmp(x, compounds(i)),compNames(:,1))); [compounds2 = compNames(NameRow(1),2)]
    if ~exist('compounds2')
        compounds2 = compNames(NameRow(1),2)
    else
        compounds2 = [compounds2; compNames(NameRow(1),2)]
    end
end

%%%% Use compounds for kegg codes and compounds2 for compounds names

names = {compounds2{:}, enzymes{:}};
network = biograph(compmat,names);
set(network, 'LayoutType', 'radial');
set(network, 'ShowArrows', 'off');
set(network, 'Scale', 1);
set(network.nodes(1:length(compounds)), 'Color', [1 0.5 0], 'size', [40 30])
set(network.nodes(1:length(compounds)), 'Shape', 'ellipse')
disp(view(network));

%%%%%%%%%%%%%%%%%%%%%%%%%%% THE END %%%%%%%%%%%%%%%%%%%%%%%%%