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 recognize 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.

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.
Vector Generalized Linear Time Series Models with an Implementation in R

Víctor Florentino Miranda Soberanis
Department of Statistics

This dissertation is submitted for the degree of Doctor of Philosophy

The University of Auckland, August 2018.

Minor corrections completed: 19 February 2019
Abstract

Since the introduction of the ARMA–class in the early 1970s many time series (TS) modelling extensions have been proposed involving linear and non–linear structures as part of a huge literature, for instance, the vector–ARMA class for multivariate TS and the ARCH–GARCH-type models for heteroskedasticity. The result has been an explosion of TS models and inference schemes having pockets of substructure but limited overriding framework. In this work, the class of vector generalized linear models (VGLMs) is shown to confer advantages towards data–types with trend and patterns that evolve over time. Specifically, we propose a new class of VGLMs, called vector generalized linear time series model (VGLTSMs), which are endowed with capabilities to handle autocorrelated data and can be thought of as multivariate generalized linear models directed towards time series. This work follows previous successful endeavours of developing VGLMs for other data types, such as categorical data, extremes, and quantile regression. The crucial VGLM ideas are constraint matrices, vector responses and covariate-specific linear predictors, and estimation by iteratively reweighted least squares and Fisher scoring. The only modification to the VGLM framework is to constrain its log–likelihood to depend on deterministic time–dependent data. We show how several popular sub-classes of TS models are accommodated as special cases of VGLTSMs, as well as new work that broadens TS modelling even more. A prominent example is cointegrated time series that is to be shown amenable to VGLTSMs by means of its ability to handle multiple responses. This work is accompanied with a new software implementation in R, called the VGAMextra package, which is available on CRAN. Its performance is compared to other software for TS analysis. Algorithmic details of its implementation, as well as many VGLTSM modelling features allowed by VGAMextra, are described here.
To Alondra
ACKNOWLEDGEMENTS

Almost four years of intensive work have elapsed before completing this thesis, where, however, I’ve pleasantly witnessed the contributions and assistance of various selfless persons and divisions who I wish to remind of as a central component in such.

First and foremost, my deepest gratitude to my main supervisor Dr. Thomas Yee who expertly led me throughout this journey flavoured with ceaseless knowledgeable words on statistics and statistical computing that kept me engaged with my research. I particularly acknowledge his efforts for proofreading this thesis in its entirety. I also wish to extend my gratitude to Dr. Ciprian Giurcaneanu, Assoc–Prof. Renate Meyer and Assoc–Prof. Ross Ihaka for their directions and comments at initial stages that helped me to narrow down the pathway to persevere with the development of this work.

Moreover, this thesis could not have been completed without the generous support from the University of Auckland (UoA) and its Department of Statistics. Many thanks to UoA who awarded me with a UNIVERSITY OF AUCKLAND DOCTORAL SCHOLARSHIP, to Mr. David Smith for giving me the opportunity and the confidence to take over GTA activities between 2014 and 2018, to Assoc–Prof. Ilze Ziedins (former Head of Department), and especially I’m very much thankful to Prof. James Curran, current Head of Department, who provided timely funding towards attending international and national conferences to present my latest results.

Last but not least, I am indebted to my wife for her unconditional love, patience and encouragement during the intensive preparations.
# Contents

## 1 Introduction

1.1 Motivation ............................................................... 1

1.1.1 An overview of this work ........................................ 2

1.1.2 An example .......................................................... 3

1.2 Background ............................................................ 8

1.2.1 ARIMA–like models .................................................. 8

1.2.1.1 Regression models with ARMA errors ...................... 8

1.2.1.2 The VARMAX model .......................................... 9

1.2.2 Unit root tests ....................................................... 9

1.2.3 Capturing the volatility dynamics of ARMA models ........ 11

1.2.4 Beyond the normality assumption ............................... 12

1.2.5 PIT plots ........................................................... 13

1.2.6 Software implementations ........................................ 14

1.3 Vector generalized linear models .................................. 16

1.3.1 Constraint matrices ............................................... 17

1.3.2 Reduced–rank VGLMs .............................................. 18

1.3.3 On VGLM estimation .............................................. 19

1.4 Aims ................................................................. 20

## 2 Vector Generalized Linear Time Series Models

2.1 Introduction ........................................................... 23

2.2 Vector generalized linear time series models ..................... 23

2.2.1 Order–(u, d, v) VGLM–ARIMAX .............................. 26

2.2.2 Order–(u, v, r, s) VGLM–ARMAX–GARCH .................. 28

2.2.3 Order(u, v) VGLM–INGARCH with interventions .......... 30

2.3 Estimation ............................................................. 34

2.3.1 The score vector and the EIMs .................................. 34

2.3.2 Inference ............................................................ 37

2.4 Forecasting ............................................................ 39

2.5 Summary .............................................................. 40
3 New VGLTSM Family Functions

3.1 Introduction ........................................... 41
3.2 On VGAMextra and VGAM ................................. 41
3.3 New family functions for VGLTSMs ....................... 43
  3.3.1 On modelling and estimation of VGLTSMs using
        VGLTSMffs ......................................... 44
  3.3.2 Initial values for VGLTSMffs ........................ 47
  3.3.3 Family functions from the class VGLM–ARIMAX ... 49
  3.3.4 Family functions from the class VGLM–ARMAX–GARCH ...................... 52
  3.3.5 VGLM.INGARCHff() and intervention analysis .......... 56
3.4 Complementary VGLTSM family functions ............... 61
  3.4.1 VARff(), a VGLTSMff for VAR–type models .......... 61
  3.4.2 ARIMAX.errors.ff() for dynamic regression ........... 63
  3.4.3 Student–t errors on ARMA modelling using
        ARMA.studentt.ff() .................................. 72
3.5 S3/S4 methods for VGLTSMffs–derived objects .......... 77
  3.5.1 The summary() generic for VGLM–ARIMAX ............. 80
3.6 Summary ................................................. 84

4 On Supplementary Modelling Features of VGLTSMs ...... 85

4.1 Introduction ............................................. 85
4.2 Constrained VGLTSMs ................................. 85
  4.2.1 Constrained VGLTSMs using the argument zero .......... 86
  4.2.2 Constrained VGLTSMs using constraints .............. 88
  4.2.3 Constrained VGLTSMs: An order(1)–AR with ARCH(1) errors 91
4.3 The model matrix X_{LM,T} ................................ 94
  4.3.1 X_{LM,T} of a regression model with ARIMA(1, 1, 1) errors .... 95
  4.3.2 The X_{LM,T} matrix of an AR(1) model with ARCH(1) errors . 95
  4.3.3 The X_{LM,T} matrix of an INGARCH(1, 0) with interventions 96
  4.3.4 The VLM matrix X_{VLM,T} ............................ 97
4.4 Supporting functions in VGAMextra ..................... 98
  4.4.1 PIT plots with VGAMextra .......................... 98
  4.4.2 Unit root tests .................................... 101
  4.4.3 Stationarity, stability and invertibility ................ 103
4.5 Some miscellaneous functions .......................... 108
4.6 Summary ................................................. 109

5 VGLTSMs Towards Cointegrated Time Series ............. 111
7.6 Summary .................................................................................... 179

8 Future Work .................................................................................. 181
  8.1 Introduction ................................................................................ 181
  8.2 Some directions for future work .................................................. 182
     8.2.1 On forecasting with VGLTSM family functions .................... 182
     8.2.2 On model selection ............................................................... 184
     8.2.3 On detecting interventions with VGLM.INGARCHff() ............. 185
     8.2.4 Some comments on VGLTSMs and on VGAMextra ............... 186

Appendices ......................................................................................... 189
  A ........................................................................................................ 191
     A.1 Summary of model fit.PM10 using VGAMextra ......................... 191
     A.2 Summary of models fit.gamlss.arima and 
        fit.gamlss.armax .................................................................... 193
     A.3 Summary of model (6.3.2) estimated with both, VGLM.INGARCH() and 
        tscount::tsglm() ..................................................................... 195
     A.4 Complementary material for model (6.3.4) ............................... 197

References ......................................................................................... 200
List of Figures

Figure 1.1.1: Residuals from (a) model (1.1.1) fitted with \texttt{forecast::auto.arima()}, (b) model (1.1.1) fitted with \texttt{VGAMextra::ARIMAX.errors.ff()}, and (c) model (1.1.2) fitted with \texttt{VGAMextra::ARIMAX.errors.ff()}. 7

Figure 3.3.1: Simulated TSC $\sim$ NB with interventions, based on (3.3.10). 60

Figure 3.4.1: Residuals from model (3.4.8) estimated with (1) \texttt{arima()} and (2) \texttt{ARIMAX.errors.ff()}. 69

Figure 3.4.2: Residuals from model (3.4.8) estimated with (1) \texttt{forecast::Arima()} and (2) \texttt{ARIMAX.errors.ff()}. 70

Figure 3.4.3: Residuals computed from (ia) \texttt{auto.arima()}, (ib) \texttt{ARIMAX.errors.ff()}, (ii) \texttt{ARMAX.ff()}, (iii) \texttt{ARMA.studentt.ff()}. 76

Figure 3.4.4: Normal QQ–plots upon residuals from models (ia)–(iii) using (ia) \texttt{auto.arima()}, (ib) \texttt{ARIMAX.errors.ff()}, (ii) \texttt{ARMAX.ff()}, (iii) \texttt{ARMA.studentt.ff()}. 77

Figure 3.4.5: Student–t QQ–plots upon residuals from models (ia)–(iii) using (ia) \texttt{auto.arima()}, (ib) \texttt{ARIMAX.errors.ff()}, (ii) \texttt{ARMAX.ff()}, (iii) \texttt{ARMA.studentt.ff()}. 78

Figure 4.4.1: PIT histograms obtained from (4.4.1) assuming data distributed as (a) negative binomial and (b) Poisson. 101

Figure 5.3.1: Simulated random walks, $x_t$ and $y_t$, from model (5.3.7). 120

Figure 5.3.2: Residuals $\hat{z}_t$ from the regression $y_t \sim x_t$ with intercept. 121

Figure 6.2.1: Time plot, PACF, ACF of PM$_{10}$, from the \texttt{ap.mx} dataset. 131

Figure 6.2.2: (a)–(c) Scatter plots of PM$_{10}$ vs. order–0 to order–2 lagged values of O$_3$; (d)–(f) PM$_{10}$ actual levels vs. one–day to three–days past readings (PM$_{10}$). 132

Figure 6.2.3: Estimated errors from model (6.2.1), computed with (a) \texttt{arima()}, and (b) \texttt{ARMAXff()}. 135

Figure 6.3.1: Number of campylobacteriosis infections in Northern Québec reported every 28 days between January 1990 and October 2010. 137
Figure 6.3.2: PIT histograms built upon model (6.3.1), computed with \texttt{PIT()} from \texttt{VGAMextra}, assuming (a) Poisson and (b) negative binomial distributions. ........................................... 138

Figure 6.3.3: Campylobacterosis infections: Fitted values from model (6.3.1) assuming (a) Poisson and (b) negative binomial response. No intervention effects. ................................................................. 139

Figure 6.3.4: Fitted values from the intervention effects model (6.3.2). .... 140

Figure 6.3.5: Daily hospital admissions due to respiratory diseases, Hong–Kong (Jan 1995–Dec 1997). ................................................................. 142

Figure 6.3.6: Series of pollutants and atmospheric variables, Hong Kong (Jan 1995–Dec 1997): (a) Ozone (ppb), (b) PM$_{10}$ (ppb), (c) NO$_2$ (ppb), (d) SO$_2$ (ppb), (e) Temperature ($^\circ$C), (f) Humidity (%). ................................................................. 143

Figure 6.3.7: Fitted values of Hong Kong hospital admissions due to respiratory diseases from model (6.3.4) assuming the response is (a) NB and (b) Poisson. ................................................................. 145

Figure 6.3.8: PIT histograms build on the estimated model (6.3.4), computed with \texttt{VGAMextra::PIT()}; (a) Poisson, (b) NB. .......... 145

Figure 7.3.1: Plot of \texttt{logfflink()} in (0,1). .......................... 171

Figure 7.3.2: Plot of $g_{\eta_i}(s)$ for: (a) $\eta_i = 0.5$, (b) $\eta_i = 1.5$, (c) $\eta_i = 3.0$, (d) $\eta_i = 6.0$. .................................................. 172

Figure 7.3.3: Some 1–parameter probability VGLM–link functions, with $0 < \theta < 1$. .......................... 173

Figure 7.4.1: Scatter plot of simulated Maxwell data (7.4.3) including (a) the fitted quantile functions from \texttt{fit.Qmodelling (VGAMextra)}), and (b) the fitted quantile curves from \texttt{fit.Qregression (VGAM)}. The quantile curves in both cases derive from vector smoothing spline fits. The points have been jittered slightly. .... 177

Figure A.4.1: Fitted values obtained with the extended–GAM model. .... 197
## List of Tables

<p>| Table 1.1.1: | AICs and MSEs from models (1.1.1) and (1.1.2) | 6 |
| Table 1.2.1: | Other conditional variance specifications for ARCH–GARCH–type models. | 12 |
| Table 1.2.2: | Popular R packages to fit univariate ARMA–GARCH–type models. | 14 |
| Table 1.2.3: | Some popular R packages to analyse time series of counts. | 15 |
| Table 1.3.1: | Some VGLM/VGAM link functions available in VGAMextra. Some are shared with VGAM. | 17 |
| Table 2.2.1: | Linear predictors handled by VGLM–ARIMAX. | 27 |
| Table 2.2.2: | Some ARIMAX–like models handled by VGLM–ARIMAX as special cases. | 27 |
| Table 2.2.3: | Linear predictors handled by VGLM–ARMAX–GARCH. | 28 |
| Table 2.2.4: | Special cases of conditional variance models handled by VGLM–ARMAX–GARCH. | 29 |
| Table 2.2.5: | One–parameter VGLM–distributions ( F ) supported by VGLM–INGARCH. | 32 |
| Table 2.2.6: | INGARCH–type models for time series of counts supported by VGLM–INGARCH: Special cases. | 33 |
| Table 2.3.1: | Score vector and EIMs of the statistical models supported by VGLM–INGARCH. | 36 |
| Table 3.2.1: | Some relevant arguments of <code>vglm()</code> | 42 |
| Table 3.3.1: | Summary of central VGLTSMffs currently available in VGAMextra. Some arguments are shown. | 43 |
| Table 3.3.2: | Some essential slots of VGLTSMffs adopted by typical VGLM–family functions (Table 18.3 of Yee, 2015). | 44 |
| Table 3.3.3: | Important arguments of VGLTSM family functions derived from VGLM–ARIMAX and VGLM–ARMAX–GARCH. | 45 |
| Table 3.3.4: | Important arguments of <code>VGLM.INGARCHff()</code>, the VGLTSMff describing the class VGLM–INGARCH. | 46 |
| Table 3.3.5: | Schemes adopted to generate initial values ( \eta^{(0)} ) in VGLTSMffs. | 47 |</p>
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.6</td>
<td>Input-types required by the argument interventions</td>
<td>58</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Complementary VGLTSMff depicted in Section 3.4</td>
<td>61</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Arguments involved with VARff()</td>
<td>62</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Central arguments of ARIMAX.errors.ff()</td>
<td>65</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Arguments handled by ARMA.studentt.ff()</td>
<td>73</td>
</tr>
<tr>
<td>3.4.5</td>
<td>AICs and BICs from models–(i)–(iii).</td>
<td>74</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Central arguments of VGAMextra::PIT()</td>
<td>99</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Arguments involved with VGAMextra::KPSS.test()</td>
<td>102</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Arguments involved with checkTS.VGAMextra()</td>
<td>105</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Miscellaneous functions in VGAMextra that assist VGLTSMffs in IRLS–Fisher scoring. Some of them are imported from other packages (details at the NAMESPACE).</td>
<td>108</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Arguments handled by ECM.EngleGran()</td>
<td>116</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Regression schemes managed by ECM.EngleGran() to estimate residuals and cointegrating vectors using resid.pattern.</td>
<td>117</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Variables included in ap.mx, collected 1\textsuperscript{st} January 2004–30\textsuperscript{th} June 2005</td>
<td>130</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Summary of model (6.2.1) fitted with ARMAXff() from VGAMextra</td>
<td>134</td>
</tr>
<tr>
<td>6.2.3</td>
<td>AICs and BICs after fitting model (6.2.1) under two approaches</td>
<td>135</td>
</tr>
<tr>
<td>6.3.1</td>
<td>AICs, BICs after fitting model (6.3.1) and modelling functions utilised.</td>
<td>138</td>
</tr>
<tr>
<td>6.3.2</td>
<td>AICs and BICs after fitting (6.3.2) with tsglm() and vglm()</td>
<td>140</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Summary of model (6.3.2) fitted with vglm() using VGLM.INGARCHff()</td>
<td>140</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Estimated percentage change in daily Hong Kong hospital admissions from model (6.3.4). Response assumed as NB.</td>
<td>144</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Expressions of the score vector for 2-parameter VGLMs considering the three compositions of the linear predictors (7.2.5)</td>
<td>155</td>
</tr>
<tr>
<td>7.2.2</td>
<td>New complementary expressions for 2-parameter VGLMs to compute the diagonal elements of the EIMs, viz. (7.2.18), for the three versions of the linear predictors (7.2.5)</td>
<td>157</td>
</tr>
<tr>
<td>7.2.3</td>
<td>New complementary expressions for 2-parameter VGLMs to compute the off-diagonal elements of the EIMs, viz. (7.2.18), for the three versions of the linear predictors (7.2.5)</td>
<td>158</td>
</tr>
<tr>
<td>7.3.1</td>
<td>New link functions for the mean of some discrete distributions.</td>
<td>168</td>
</tr>
</tbody>
</table>
Table 7.3.2: New VGLM–links for the mean of some continuous 1–parameter distributions. ................................. 169

Table 7.3.3: Mean–links currently available in VGAMextra ............................... 170

Table 7.4.1: New link functions for the quantiles of some 1–parameter distributions. ........................................ 174

Table 7.4.2: New link functions for the quantiles of 1–parameter distributions: Its inverse and names in VGAMextra. “Approximate” means that Newton–Raphson or bisection is used to approximate the inverse. ......................................................... 176

Table 7.4.3: Data coverage from quantile–modelling using VGAMextra (QM–VGAMextra), and quantile–regression from VGAM (QR–VGAM), after fitting (7.4.3). ................................................................. 178

Table 7.5.1: Other VGLMffs in VGAMextra. .................................................. 179
# Table of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike information criterion</td>
</tr>
<tr>
<td>AICc</td>
<td>Corrected AIC</td>
</tr>
<tr>
<td>ARIMA</td>
<td>Autoregressive integrated moving average (Section 1.2)</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive moving average (Section 1.2)</td>
</tr>
<tr>
<td>ARMAX</td>
<td>ARMA model with exogenous terms</td>
</tr>
<tr>
<td>ARX</td>
<td>Autoregressive model with exogenous terms</td>
</tr>
<tr>
<td>BAMLSS</td>
<td>Bayesian additive models for location, scale and shape</td>
</tr>
<tr>
<td>BHHH</td>
<td>The Bernt, Hall, Hall, Hausman iterative algorithm</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian information criteria</td>
</tr>
<tr>
<td>CRAN</td>
<td>Comprehensive R Archive Network (<a href="http://cran.R-project.org">http://cran.R-project.org</a>)</td>
</tr>
<tr>
<td>CTS</td>
<td>Cointegrated time series</td>
</tr>
<tr>
<td>DGP</td>
<td>Data generating process</td>
</tr>
<tr>
<td>DRM</td>
<td>Dynamic regression models</td>
</tr>
<tr>
<td>ECM</td>
<td>Error correction model</td>
</tr>
<tr>
<td>EM–algorithm</td>
<td>Expectation–maximization algorithm</td>
</tr>
<tr>
<td>GARCH</td>
<td>Generalized autoregressive conditional heteroskedasticity (Section 1.2)</td>
</tr>
<tr>
<td>GAM</td>
<td>Generalized additive model</td>
</tr>
<tr>
<td>GARMA</td>
<td>Generalized autoregressive moving average</td>
</tr>
<tr>
<td>GLM</td>
<td>Generalized linear model</td>
</tr>
<tr>
<td>i.i.d</td>
<td>Independent and identically distributed</td>
</tr>
<tr>
<td>INGARCH</td>
<td>Integer–valued GARCH (Section 1.2, Eq. (1.2.11))</td>
</tr>
<tr>
<td>LM</td>
<td>Linear model (Section 2.2 in Yee (2015))</td>
</tr>
<tr>
<td>LSS</td>
<td>Location, scale, and shape</td>
</tr>
<tr>
<td>MLE</td>
<td>Maximum likelihood estimation or estimator</td>
</tr>
<tr>
<td>MNoise</td>
<td>Multivariate noise</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean squared error (Euclidean norm)</td>
</tr>
<tr>
<td>MTS</td>
<td>Multivariate time series</td>
</tr>
<tr>
<td>MWN</td>
<td>Multivariate white noise</td>
</tr>
<tr>
<td>NB</td>
<td>Negative binomial</td>
</tr>
<tr>
<td>NB-C(_1)-H</td>
<td>NB distribution with canonical link and (k) known</td>
</tr>
<tr>
<td>NB-C(_2)-2</td>
<td>NB distribution with canonical link and (k) unknown</td>
</tr>
<tr>
<td>OIM</td>
<td>Observed information matrix</td>
</tr>
<tr>
<td>OLS</td>
<td>Ordinary least squares</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OOS</td>
<td>Object oriented system</td>
</tr>
<tr>
<td>OOP</td>
<td>Object oriented programming</td>
</tr>
<tr>
<td>ppb</td>
<td>parts per billion</td>
</tr>
<tr>
<td>QMLE</td>
<td>Quasi–MLE</td>
</tr>
<tr>
<td>QQ–plot</td>
<td>Quantile–quantile plot</td>
</tr>
<tr>
<td>RRR</td>
<td>Reduced rank regression</td>
</tr>
<tr>
<td>RR–VGLM</td>
<td>Reduced–rank vector generalized linear model (Section 1.3.2)</td>
</tr>
<tr>
<td>ResSS</td>
<td>Residual sum of squares</td>
</tr>
<tr>
<td>SE</td>
<td>Standard error</td>
</tr>
<tr>
<td>TSC</td>
<td>Time series of counts</td>
</tr>
<tr>
<td>URT</td>
<td>Unit root test (Sections 1.2.2, 4.4.2)</td>
</tr>
<tr>
<td>VLM</td>
<td>Vector linear model. Eq. (3.22), Fig. 1.2, in <em>Yee</em> (2015).</td>
</tr>
<tr>
<td>VGAM</td>
<td>Vector generalized additive model (Section 1.3)</td>
</tr>
<tr>
<td>VGLM</td>
<td>Vector generalized linear model (Section 1.3)</td>
</tr>
<tr>
<td>VGLMff</td>
<td>VGLM family function</td>
</tr>
<tr>
<td>VGLM–ARIMA</td>
<td>Vector generalized linear model–ARIMA class (Section 2.2.1)</td>
</tr>
<tr>
<td>VGLM–ARMAX–GARCH</td>
<td>Vector generalized linear model–ARMAX–GARCH class (Section 2.2.2)</td>
</tr>
<tr>
<td>VGLM–INGARCH</td>
<td>Vector generalized linear model–INGARCH class (Section 2.2.3)</td>
</tr>
<tr>
<td>VGLTSM</td>
<td>Vector generalized linear models time series models (Section 2.2)</td>
</tr>
<tr>
<td>VGLTSMff</td>
<td>VGLTSM family function (Chapter 3)</td>
</tr>
<tr>
<td>VARX</td>
<td>Vector autoregressive model with exogenous terms (Section 1.3.2)</td>
</tr>
<tr>
<td>VECM</td>
<td>Vector ECM</td>
</tr>
<tr>
<td>WN</td>
<td>White noise</td>
</tr>
</tbody>
</table>

Many R functions from different packages are cited in this work, and the following convention is used: `package::function-name()`, e.g., `VGAM::binormal()`. Cited functions from my VGAMextra package do not carry in general the package name, e.g, `ARXff()`, but sometimes `VGAMextra::ARXff()` is used for emphasis.
# Glossary

Summary of some notation used throughout this thesis.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{N}_0$</td>
<td>Set of all nonnegative integers, $0(1)\infty$</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>Set of real numbers $(-\infty, \infty)$</td>
</tr>
<tr>
<td>$\mathbb{Z}_{\leq 0}$</td>
<td>Non–positive integers</td>
</tr>
<tr>
<td>$\Gamma$</td>
<td>Gamma function, $\Gamma(t) = \int_0^\infty x^{t-1} \exp(-x)dx$, for $t \in \mathbb{R}\setminus\mathbb{Z}_{\leq 0}$</td>
</tr>
<tr>
<td>$\text{Be}$</td>
<td>Beta function, $\text{Be}(x, y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x + y)}$, $x, y &gt; 0$</td>
</tr>
<tr>
<td>$\zeta$</td>
<td>Riemann zeta function, $\zeta(\theta) = \sum_{n=1}^{\infty} \frac{1}{n^\theta}$, $\theta &gt; 1$</td>
</tr>
<tr>
<td>$\psi(\cdot)$</td>
<td>Digamma function, that is, the first derivative of the log–gamma function</td>
</tr>
<tr>
<td>$\iff$</td>
<td>If and only if, i.e., a necessary and sufficient condition, $\iff$</td>
</tr>
<tr>
<td>$\mathcal{N}_2$</td>
<td>Bivariate normal distribution</td>
</tr>
<tr>
<td>$A_{m\times n}$</td>
<td>An order–$(m \times n)$ matrix $A$</td>
</tr>
<tr>
<td>$\otimes$</td>
<td>Kronecker product, $A \otimes B = [(a_{ij} \cdot B)]<em>{mp\times nq}$, where $A</em>{m\times n}$ and $B_{p\times q}$</td>
</tr>
<tr>
<td>$\text{erf}(\cdot)$</td>
<td>Error function, $\text{erf}(x) = \left(\frac{2}{\sqrt{\pi}}\right) \int_0^x e^{-t^2} dt$</td>
</tr>
<tr>
<td>$\text{diag}(a_1, \ldots, a_p)$</td>
<td>A $p \times p$ diagonal matrix with diagonal elements $a_1, \ldots, a_p$</td>
</tr>
<tr>
<td>$I_M$</td>
<td>The order–$M$ identity matrix</td>
</tr>
<tr>
<td>$\Phi_t$</td>
<td>Information set at time $t$ of a process ${Y_t}$</td>
</tr>
<tr>
<td>$B$</td>
<td>Backshift operator: $BY_t = Y_{t-1}$</td>
</tr>
<tr>
<td>$\nabla^d$</td>
<td>Order–$d$ difference operator: $\nabla^d(\cdot) = (1 - B)^d(\cdot)$</td>
</tr>
<tr>
<td>$\nabla$</td>
<td>First difference operator: $\nabla(\cdot) = (1 - B)(\cdot)$</td>
</tr>
<tr>
<td>$I(d)$</td>
<td>Integrated of order $d$</td>
</tr>
<tr>
<td>$CI(b, d)$</td>
<td>Cointegrated of order $b, d$</td>
</tr>
</tbody>
</table>
Summary of further notation used throughout this thesis.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(b)c</td>
<td>The set ${a, a + b, \ldots, c}$, $a, b, c \in \mathbb{R}$, and $a &lt; c$</td>
</tr>
<tr>
<td>Re($x$), Im($x$)</td>
<td>Real and imaginary parts of the complex $x$</td>
</tr>
<tr>
<td>sup</td>
<td>Supremum</td>
</tr>
<tr>
<td>I(·)</td>
<td>The indicator function</td>
</tr>
<tr>
<td>tr()</td>
<td>Trace</td>
</tr>
<tr>
<td>◦</td>
<td>Hadamard product (element by element)</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

Since the introduction of the seminal ARIMA–class of models (Box and Jenkins, 1970), the analysis of time series (TS) has been an area of much research activity. In terms of modelling, estimation and forecasting, many methodologies have been proposed that have blossomed into neighbouring topics. Among this bewildering structure, regression methods for TS analysis have largely benefited from the generalized linear models (GLM) methodology that have conferred advantages beyond the ordinary assumption of normal responses, and further, on allowing random explanatory variables in the process generated by the conditional mean \( \mathbb{E}(Y_t | \Phi_{t-1}) \), see, e.g., Kedem and Fokianos (2002).

The class of dynamic regression models (DRMs) are a well–known instance derived from regression techniques that may be fitted to time series data. However, the influence of external factors here is ordinarily assessed through the conditional mean model, impeding a formal analysis of their potential impact on the error’s structure, e.g., the error’s variance model. Further choices are hierarchical time series and the well–known class of volatility models with covariates, which is a very active area in financial markets. While the former choice offers a consistent approach to model interactions and correlations between a number of series, it ignores the potential impact of other factors on the error structure, with potential side effects, e.g., influencing the forecasting prediction intervals (see, e.g. Hyndman et al., 2011). The latter has increasingly gained popularity, but still confined to the GARCH–class, see, e.g., Tsung-Han and Yu-Pin (2013); Francq and Quyen-Thieu (2018).

This work proposes an extension of the GLM–type regression models for TS analysis using vector generalized linear models (VGLMs). VGLMs are supported by the VGAM R package (R Core Team, 2017), which is an implementation of 6 major classes of statistical models for regression analysis (Yee, 2015). Fundamentally, this thesis is
concerned with the VGLM framework and how its inferential substance ought to be modified to accommodate time series. This work aims to demonstrate how the resulting VGLM–type class of models for TS confers several advantages that have been encountered previously in time series analysis but with much less an overriding framework. Specifically:

- Handling responses beyond the 1–parameter GLM family of distributions.
- Accommodating multiple and multivariate time series, connecting them with time–dependent covariate–specific linear predictors via parameter constraints.
- Exploring the potential influence of external factors intervening with the volatility structure of the series, beyond the ordinary ARCH–GARCH variants.
- Investigating how the error’s homoskedasticity as well as the assumptions of stationarity and invertibility of ARIMA–like processes may be influenced by time–dependent factors.

Another outcome derived from this work is a companion software implementation in R, called the VGAMextra package, which is available on CRAN (https://cran.r-project.org/package=VGAMextra). To date, the CRAN repository of R packages encompasses a range of software for the modelling, analysis and forecasting from TS, where FitARMA, forecast, and tseries, are just a few examples. However, numerous packages in this context essentially call the function optim() for internal optimization procedures, e.g., the popular arima() to fit ARIMA–like models, or the packages rugarch (Ghalanos, 2014) to estimate, simulate an forecast ARFIMA models with GARCH errors, or tsDyn (Fabio Di Narzo et al., 2009), handling non–linear autoregressive models and vector error–correction models to explore threshold cointegration. The present work aims to demonstrate better performance than these.

1.1.1 An overview of this work

This thesis is primarily focused on modelling and estimation, and attempts to simultaneously address the above features using VGLMs. In short, the main purpose of this thesis is two–fold:

a) It proposes a new class of VGLMs directed towards TS data, called vector generalized linear time series models (VGLTSMs). Following the VGLM mainstream, theory and software have been developed for a new basic modelling and inference framework for time series, including time series of counts. Estimated by MLE using Fisher scoring, VGLTSM are to be shown to encompass popular univariate and multivariate platforms for TS analysis. The central VGLM attributes
are inherited by VGLTSMs: constraint matrices, vector responses outside the 1–parameter GLM family, and covariate–specific linear predictors, whose potential usefulness in a number of application areas, e.g., cointegrated time series, is demonstrated in this thesis.

b) In addition, this thesis introduces VGAMextra, the software implementation for VGLTSMs. The package comprises functions and methods to make VGLTSMs operational, including the set of statistical models describing VGLTSMs (called VGLTSM family functions or VGLTSMff), as well as S4 object oriented programming (OOP) routines dispatching \texttt{summary} and \texttt{show} methods for VGLTSMs.

However, VGLTSMs are even broader. They possess infrastructure to cope with other relevant techniques for TS analysis, e.g., automatic forecasting, systems of cointegrated time series, non–parametric and parameter–driven TS models, and errors with distributions outside the exponential family. While many of such subjects are still in development for VGLTSMs (and are not presented in this work), Chapter 8 gives some grounds of this work and discusses future endeavours in this respect.

1.1.2 An example

As motivation, we present the results of a short example on how flexible and suitable the VGLTSM framework can be for analyzing TS using dynamic regression models (DRMs). The data utilized here is \texttt{WWWusage} (Durbin and Koopman, 2001) from \texttt{datasets}, which comprises a series of 100 observations ($n = 100$), each one giving the number of users connected to an internet server every minute. For illustration purposes, an artificial covariate, $x_{t,2} \overset{i.i.d}{\sim} \text{Unif}(1, 5)$, is generated and incorporated in the analysis, assuming (the usual) Gaussian noise. The VGLTSM–schemes adopted here as well as the VGLTSM family functions to estimate such are presented at an introductory level. Deeper details are pursued in Chapters 2 and 3.

(1) We start by selecting an initial modelling structure that describes fairly the DGP assumed in this example. For this, we use \texttt{forecast::auto.arima()} (Hyndman and Khandakar, 2008), a wrapper function of \texttt{stat::arima()} which estimates and returns the best ARIMA model according to either the AIC, AICc, or BIC criteria when no covariates are included in the analysis. In the presence of covariates, like in this example with the inclusion of $x_{t,2}$, \texttt{forecast::auto.arima()} will fit a linear regression model $y_t \sim x_{t,2}$ with the ARIMA($p, d, q$) structure imposed on the error term, $u_t$, also referred to as \textit{regression with ARIMA errors}. The results from \texttt{auto.arima()} are shown in the following output, saved as \texttt{fit.auto.arima}.
Chapter 1. Introduction

> set.seed(1); n <- 100
> ts.data <- data.frame(y = WWWusage, x2 = runif(n, 1, 5))
> # Fitting a regression model with ARIMA(p, d, q) errors.
> (fit.auto.arima <- with(ts.data, auto.arima(y, xreg = x2)))

Series: y
Regression with ARIMA(1,1,1) errors

Coefficients:
    ar1   ma1  xreg
coefficients: 0.650 0.526 0.006
              s.e. 0.084 0.089 0.087
sigma^2 estimated as 10.1: log likelihood=-254.15
AIC=516.29  AICc=516.72  BIC=526.67

The selected model for the error series is an ARIMA(1, 1, 1), which implies that the regression is performed over $\nabla y_t|_{\Phi_{t-1}} \sim \nabla x_{t,2}$, rather than $y_t|_{\Phi_{t-1}} \sim x_{t,2}$. The full structure of the model fitted by `auto.arima()` is:

$$
(1 - \theta B) \nabla u_t|_{\Phi_{t-1}} = (1 + \phi B) z_t, \\
\nabla u_t|_{\Phi_{t-1}} = \nabla y_t|_{\Phi_{t-1}} - (\beta_0 + \beta_1 \nabla x_{t,2}),
$$

(1.1.1)

with $z_t \overset{i.i.d.}{\sim} N(0, \sigma_z^2)$, where the coefficients $\theta = (\theta, \phi, \beta_0, \beta_1)^T$ have been estimated.

(2) Now, (1.1.1) is fitted to the data using VGLTSMs. In VGAMextra, dynamic regression models, such as (1.1.1), are handled by the VGLTSM family function `ARIMAX.errors.ff()`. As a sub-class of VGLMs, this function is called by the modelling function `vglm()`, the central engine for fitting VGLMs (details are given in Section 3.2).

Unlike `auto.arima()` which makes use of gradient algorithms for optimization via `optim()`, `vglm()` performs MLE using Fisher scoring for this. The results are the following. Note that `diffcovs = TRUE` (for compatibility with `auto.arima()`), which means that any explanatory variables included are differenced, $x_{t,2}$ in this case, and then embedded in the regression model $\nabla y_t|_{\Phi_{t-1}}$. 
1.1. Motivation

> **## A regression model on \( E(y_t | x_\{t - 2\}) \) with ARIMA(1,1,1) errors.**
> **## 'diffcovs' = TRUE, i.e., the 1st difference on the covariates**
> **## is considered.**
> fit.vgam1 <- vglm(y ~ x2, ARIMAX.errors.ff(order = c(1, 1, 1),
> diffCovs = TRUE, xLag = 0,
> include.int = FALSE),
> data = ts.data, trace = TRUE)

VGLM linear loop 1: loglikelihood = -277.59146
VGLM linear loop 2: loglikelihood = -258.35919
VGLM linear loop 3: loglikelihood = -253.49841
VGLM linear loop 4: loglikelihood = -253.24354
VGLM linear loop 5: loglikelihood = -253.24288
VGLM linear loop 6: loglikelihood = -253.24288

> t(coef(fit.vgam1, matrix = TRUE))

(Intercept) Diffx2 ARcoeff1 MAcoeff1
mean1 0.0000 0.05572 0.64682 0.54628
loglink(var1) 2.2781 0.00000 0.00000 0.00000

The estimated coefficients of the regression model and the ARIMA(1,1,1) structures largely conform with the results from `auto.arima()`. In addition, the residuals variance is also estimated: \( \hat{\sigma}_z^2 = 9.76 \). More details on `ARIMAX.errors.ff()` are given in Section (3.4.2).

(3) Optionally, it may be of interest to incorporate lagged values of the covariate differences in the regression model, say lag 1. `ARIMAX.errors.ff()` accommodates such additions via the argument `xLag`. The model referred is of the form:

\[
(1 - \theta B) \nabla u_t|_{t-1} = (1 + \phi B) z_t \\
\nabla u_t|_{t-1} = \nabla y_t|_{t-1} - (\beta_0 + \beta_1 \nabla x_{t-2} + \beta_2 \nabla x_{t-1,2}),
\]

(1.1.2)

and can be fitted to `WWWusage` with the following code, saved as `fit.vgam2`. Note that `xLag = 1` (the default is `xLag = 0`), which incorporates the series \( \nabla x_{t-1,2} \) in the analysis.
# Chapter 1. Introduction

Table 1.1.1. AICs and MSEs from models (1.1.1) and (1.1.2)

<table>
<thead>
<tr>
<th>Case</th>
<th>Modelling function</th>
<th>Family function</th>
<th>AIC</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td><code>auto.arima()</code></td>
<td></td>
<td>516.29</td>
<td>10.1</td>
</tr>
<tr>
<td>(2)</td>
<td><code>vglm()</code></td>
<td><code>ARIMA.errors.ff(xLag = 0)</code></td>
<td>514.49</td>
<td>10.06</td>
</tr>
<tr>
<td>(3)</td>
<td><code>vglm()</code></td>
<td><code>ARIMA.errors.ff(xLag = 1)</code></td>
<td>516.33</td>
<td>10.15</td>
</tr>
</tbody>
</table>

```r
> ## VGLMs handling an LM for $E(y_t | x_t)$, plus an ARIMA(1, 1, 1)
> ## for the errors
> fit.vgam2 <- vglm(y ~ x2, ARIMAX.errors.ff(order = c(1, 1, 1),
>                   diffCovs = TRUE, xLag = 1,
>                   include.int = FALSE),
>                   data = ts.data)
> t(coef(fit.vgam2, matrix = TRUE))  ## The estimated coefficients

   (Intercept) Diffx2 Diffx2Lag1 ARcoeff1 MAcoeff1
mean1     0.0000  -0.0051363 -0.10259  0.64819  0.54135
loglink(var1) 2.2766   0.0000000  0.00000   0.00000   0.00000
```

To compare the relative quality of the fitted models, viz. `fit.auto.arima`, `fit.vgam1`, and `fit.vgam2`, the mean squared error (MSE) as well as the AICs are shown in Table 1.1.1, while Figure 1.1.1 shows the residuals that have been retrieved using the corresponding S3/S4 methods implemented for `auto.arima()` and `vglm()`. The results from such analyses are by no means conclusive. The statistical frameworks between Models (1)–(2) and Model (3), as well as the underlying optimization algorithms, somewhat differ. However, the differences in AICs and MSEs between Model (1) and Model (2), and between Model (1) and Model (3) highly suggest that VGLTSMs performs best with the autocorrelation pattern of WWWusage, in presence of other random factor potentially influencing the series.

Nevertheless, VGLTSMs still have a number of demerits. For instance, VGLTSMs do not yet handle the important features of non–seasonal terms, nor routines for determining the number of differences required to make a time series stationarity (of high relevance in model selection) for the ARIMA class, as with `auto.arima()`. At present, in addition, VGLTSMs lack suitable VGLM–based methods for forecasting, as well as procedures for model selection in univariate and high dimensional TS. Cointegrated time series (CTS) are amenable to VGLMs, although, to date, we have developed theory (and software) for the bivariate case only via error correction models. This subject is surveyed in Chapter 5.
Figure 1.1.1. Residuals from (a) model (1.1.1) fitted with \texttt{forecast::auto.arima()}, (b) model (1.1.1) fitted with \texttt{VGAMextra::ARIMAX.errors.ff()}, and (c) model (1.1.2) fitted with \texttt{VGAMextra::ARIMAX.errors.ff()}. Accordingly, \texttt{VGAMextra} still requires further complementing work to be integrated into the current methods on TS modelling and estimation, including automatic model selection, and forecasting methods based on VGLTSMs. Moreover, VGLMs are equipped with family functions beyond GLMs that can be extended to handle time series data by means of the flexible VGLM modelling framework. This subject signifies further potential directions to develop VGLTSMs, and to demonstrate the basics behind this idea we introduce along with this work a first contribution in this line: the family function \texttt{VGAMextra::ARMA.studentt.ff()}, the first function in \texttt{R} to my knowledge that fits ARMA–type models with Student–\(t\) errors by MLE. Details are given in Section 3.4.3.

In the following, some historical details towards TS models are provided as part of the background of this work. In addition, some other related topics are outlined here given its relevance in this and future work that following chapters make reference to. It must be noted that some of them have been only partially developed, however this
is introduced as initial work which will be linked to further future enhancements of VGAMextra, for instance, other options for unit root tests, that are planned to take part in upcoming work on automatic model selection using VGLTSMs.

1.2 Background

1.2.1 ARIMA–like models

Starting with ARIMA–like models, these structures for TS modelling are the iconic linear parametrization to accommodate weak–stationary series by determining the order of differencing needed to stationarize the series. In general, the order–\((p, d, q)\)(\(P, D, Q\)) ARIMA class is widely used for forecasting under trend features and seasonal effects, assuming the series become stable after differencing. Typically, ARIMAs are decomposed in the autoregressive \((p, P)\), integrated \((d, D)\), and the moving average \((q, Q)\) components. Identification and estimation techniques for ARIMAs were proposed by Box and Jenkins (1970).

The order–\((p, q)\) ARMAs are a sub–class of ARIMAs characterizing the mean of the process conditional on exogenous–based past information. These models rely on the assumption of weakly–stationary white noise \((\mathcal{WN})\), say \(\varepsilon_t\). Overall, an ARMA process \(\{Y_t|\Phi_{t-1}; t \in \mathbb{Z}\}\) with unknown parameters \(\theta = (\theta_1, \ldots, \theta_u)^T\), \(\phi = (\phi_1, \ldots, \phi_v)^T\), \(\mu^*\), and \(\sigma^2_{\varepsilon}\) derives from the structure

\[
\begin{align*}
\varepsilon_t &\sim \mathcal{WN}(0, \sigma^2_{\varepsilon}), \\
Y_t|\Phi_{t-1} &\sim \mathcal{F}(\mu_t|\phi_{t-1}, \sigma^2_{\varepsilon}), \\
y_t &\equiv \mu_t|\phi_{t-1} + \varepsilon_t, \\
\mu_t|\phi_{t-1} &\equiv \mu^* + \theta^T y_{t-u} + \phi^T \varepsilon_{t-v},
\end{align*}
\] (1.2.1)

with \(y_{t-u} = (y_{t-u}, \ldots, y_{t-1})^T\), \(\varepsilon_{t-v} = (\varepsilon_{t-v}, \ldots, \varepsilon_{t-1})^T\). The symbol \(\Phi_t\) denotes the information set at time \(t\) of the process \(\{Y_t\}\). The distributional assumption \(\mathcal{F}\) is usually adopted as Gaussian (white noise), although a few variants were initially explored, e.g., Student–\(t\) errors (Engle and Bollerslev, 1986; Hsieh, 1989; Nelson, 1992).

1.2.1.1 Regression models with ARMA errors

A prominent extension of the ARMA family is the ARMAX class, sometimes referred as dynamic regression models (Pankratz, 1991). ARMAXs allow independent variables in the conditional mean model, that is \(\mu_t|\phi_{t-1}\) in (1.2.1), acting as covariates. For the ease of interpretation, ARMAXs are expressed as simple regression models where the ARMA structure is imposed on the errors, say \(\eta_t|\phi_{t-1}\). The general expression of the
order\((p, q)\)–ARMAX modes with \(k\) covariates, say \(x_t = (x_{t,1}, \ldots, x_{t,k})^T\), is

\[
\left(1 - \sum_{i=1}^{p} \theta_i\right) \eta_{t|\Phi_{t-1}} = \left(1 + \sum_{j=1}^{q} \phi_j\right) z_t, \tag{1.2.2}
\]

\[\eta_{t|\Phi_{t-1}} = y_{t|\Phi_{t-1}} - (\beta_0 + \beta_1 x_{t,1} + \cdots + \beta_k x_{t,k}).\]

This model is also known as regression with ARMA errors. For ARIMA errors with differencing order \(d\), one simply incorporates the differencing operator, \(\nabla = (1 - B)\):

\[
\left(1 - \sum_{i=1}^{p} \theta_i\right) \nabla^d \eta_{t|\Phi_{t-1}} = \left(1 + \sum_{j=1}^{q} \phi_j\right) z_t, \tag{1.2.3}
\]

\[\nabla^d \eta_{t|\Phi_{t-1}} = \nabla^d y_{t|\Phi_{t-1}} - (\beta_0 + \beta_1 \nabla^d x_{t,1} + \cdots + \beta_k \nabla^d x_{t,k}),\]

where \(z_t\) is still assumed as white noise. In practice, \(x_t\) may involve dummy variables allowing calendar variation effect, or other economic exogenous variables influencing the response.

1.2.1.2 The VARMAX model

The ARMAX idea has been extended to multivariate time series. The vector autoregressive moving average model with exogenous variables (VARMAX) is example of such. Let \(y_t\) be an \(r\)–dimensional time series vector. The VARMAX\((p, q, s)\) model is defined as

\[
y_{t|\Phi_{t-1}} = \sum_{k=0}^{s} \Theta_k x_{t-k} + \sum_{i=1}^{p} \Phi_i y_{t-i|\Phi_{t-i-1}} + \varepsilon_t + \sum_{j=1}^{q} \Xi_j \varepsilon_{t-j}, \tag{1.2.4}
\]

with \(\varepsilon_t\) being an \(r\)–dimensional vector white noise process, and \(\Theta_{r \times k}, \Phi_{r \times r},\) and \(\Xi_{r \times r}\) being matrices of coefficients.

1.2.2 Unit root tests

Unit root tests (URT) have received particular attention since early the 1980s essentially to check whether a time series possesses a unit root, allowing to verify non–stationarity. Since then, a number of variants have been proposed, e.g., the Dickey–Fuller test (Dickey and Fuller, 1979), Phillips–like tests (Phillips, 1987; Phillips and Perron, 1988), or Said and Dickey (1984).

One of the well–studied URTs is the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) proposed by Kwiatkowski et al. (1992). Unlike the Dickey–Fuller or the Phillips versions testing the null hypothesis of a unit root, Kwiatkowski et al. (1992) proposes methodology to test the null hypothesis of stationarity against the alternative of a unit
root. More precisely, for a time series, say \( \{y_t; t = 1, \ldots, T\} \), the KPSS approach proposes \( y_t \) to be decomposed as the sum of a deterministic trend, a random walk, and a stationary error, in the form

\[
y_t = \rho t + \delta_t + \varepsilon_t. \tag{1.2.5}
\]

Here, \( \{\varepsilon_t\} \) is a stationary sequence of disturbances, and \( \{\delta_t\} \) is the random walk \( \delta_t = \delta_{t-1} + z_t \) with \( z_t \sim i.i.d. (0, \sigma_z^2) \).

The idea behind the KPSS test is to determine whether the random walk \( \{\delta_t\} \) has zero variance, hence, \( H_0: \sigma_z^2 = 0 \), since \( \{\varepsilon_t\} \) is assumed to be stationary. In this way, the following two options are available from (1.2.5):

1. \( \rho \neq 0 \), that is, testing the null hypothesis of \( \{y_t\} \) is stationary around a trend (trend–stationary), then (1.2.5) holds, or

2. the special case when \( \rho = 0 \), then testing the null hypothesis of \( \{y_t\} \) is stationary around a level \( \delta_0 \) (rather than around a trend).

The test–statistic, denoted as \( \kappa \), has the form

\[
\kappa = T^{-2} \sum_t \hat{S}_t^2 / s^2(t), \tag{1.2.6}
\]

where \( \hat{S}_t \) is the (estimated) cumulative sum \( \hat{S}_t = \sum_j \hat{\varepsilon}_j \), and \( s^2(t) \) are the weighted sums

\[
s^2(t) = T^{-1} \sum_{t=1}^T \hat{\varepsilon}^2 + (2/T) \sum_{j=1}^t f(j, \iota) \sum_{t=j+1}^T \hat{\varepsilon}_t \hat{\varepsilon}_{t-j}. \tag{1.2.7}
\]

The threshold \( \iota \) is called lag–truncation parameter, which must be pre–specified, whereas \( f(j, \iota) \) is the Newey–West correction factor (Newey and West, 1987), given by \( f(j, \iota) = 1 - j/(\iota + 1) \). There is no standard methodology to select an appropriate value for \( \iota \), however, satisfactory results have been found for \( \iota \) proportional to \( T^{1/2} \). Empirically, this parameter may be suggested by the problem in turn, and should be large enough to approximate the true dynamic behaviour of the series, see, e.g., Andrews (1991) for a discussion on this.

The errors in (1.2.7) are estimated according to (1.2.5). That is,

a) from the regression \( y_t \sim 1 + t \), to test the null hypothesis of trend–stationarity, or

b) from \( y_t \sim 1 \), to test the null hypothesis of level–stationarity.

The asymptotic distribution of (1.2.6) is largely discussed in Kwiatkowski et al. (1992), assuming \( \{z_t\} \) and \( \{\varepsilon_t\} \) are normal. A software implementation of the KPSS test is \texttt{kpss.test()} , available in \texttt{tseries}, which is utilized by, e.g., \texttt{auto.arima()} in model selection concerning the ARIMA–class.
This work on VGLTSMs has seen URTs developed at an early stage where the KPSS approach has been particularly adopted. Its implementation in VGAMextra is fully described in Section 4.4.2 and its advantages over \texttt{tseries::kpss.test()} are to be demonstrated.

### 1.2.3 Capturing the volatility dynamics of ARMA models

VGLTSMs have also been extended to handle ARCH–GARCH models as special cases. All the details regarding the implemented outcomes will be given in Chapter 3. This section gives some background of this work.

Following from Section 1.2.1, the ARIMA class, as a consequence of its restriction to homoskedastic errors, has shown to be inadequate on different subjects such as econometric forecasting to predict future values, or modelling inflation rate, which has been found to significantly vary over time (McNees, 1979; Engle, 1982; Engle and Kraft, 1983). ARCH–GARCH-like modelling options over the conditional forecast variance have then emerged since early 1980’s based on the influential work of Engle (1982) and Bollerslev (1986) with i.i.d standard shocks as stochastic component widely assumed as standard Gaussian noise. The linear ARMA($p$, $q$) process with order–($r$, $s$) GARCH errors has the general form:

$$
z_t \overset{i.i.d.}{\sim} N(0, 1),$$

$$Y_t|\phi_{t-1} \sim N(\mu_t|\phi_{t-1}, \sigma^2_t|\phi_{t-1}),$$

$$y_t = \mu_t|\phi_{t-1} + z_t \cdot h(\varepsilon_{t-r}, \sigma_{t-s}; \phi_{t-1}),$$

$$\mu_t|\phi_{t-1} = \mu^* + \theta^T y_{t-u} + \Phi^T \varepsilon_{t-v},$$

where clearly,

$$\text{Var}(Y_t|\phi_{t-1}) = \sigma^2_t|\phi_{t-1} = h(\varepsilon_{t-r}, \sigma_{t-s}; \phi_{t-1}).$$

Here, $h$ is an unknown real-valued positive function of past (deterministic) information

$$\varepsilon_{t-r} = (\varepsilon_{t-1}, \ldots, \varepsilon_{t-r})^T \quad \text{and} \quad \sigma_{t-s} = (\sigma_{t-1}, \ldots, \sigma_{t-s})^T.$$

Modelling alternatives for the function $h$ has a rich theory. Hansen and Lunde (2005), for instance, describe nearly 330 proposals, and many others may have been presented since then. The parametric linear GARCH class introduced by Bollerslev (1986) may be the classical scheme of (1.2.8), characterized by

$$h(\varepsilon_r, \sigma_s; \Phi_{t-1}) = \omega + \alpha^T \varepsilon^2_{t-r} + \gamma^T \sigma^2_{t-s},$$

with additional parameters $\alpha = (\alpha_1, \ldots, \alpha_r)^T$ and $\gamma = (\gamma_1, \ldots, \gamma_s)^T$. 
Table 1.2.1. Other conditional variance specifications for ARCH–GARCH–type models.

<table>
<thead>
<tr>
<th>ARCH–GARCH Model</th>
<th>Conditional variance (function $h$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalized–ARCH (GARCH, 1982)</td>
<td>$\sigma_t^2 = \omega + \sum_{u=1}^{r} \alpha_u \varepsilon_{t-u}^2 + \sum_{v=1}^{s} \gamma_v \sigma_{t-v}^2$</td>
</tr>
<tr>
<td>Integrated GARCH (IGARCH, 1986)</td>
<td>$\sigma_t^2 = \omega + \sum_{u=1}^{r} \alpha_u \varepsilon_{t-u}^2 + \sum_{v=1}^{s} \gamma_v \sigma_{t-v}^2$ subject to $\sum_u \alpha_u + \sum_v \gamma_v = 1$</td>
</tr>
<tr>
<td>Multiplicative–GARCH (1986 – 1987) log</td>
<td>$\log \sigma_t^2 = \omega + \sum_{u=1}^{p} \alpha_u \log \varepsilon_{t-u}^2 + \sum_{v=1}^{q} \gamma_v \log \sigma_{t-v}^2$</td>
</tr>
<tr>
<td>General–functional ARCH (NARCH, 1992) ‡</td>
<td>$\sigma_t^2 = \left[ \phi_0(\sigma_t^2)^\delta + \phi_1(\varepsilon_{t-1}^2)^\delta + \cdots + \phi_p(\varepsilon_{t-p}^2)^\delta \right]^{1/\delta}$, where $\sigma^2 = \text{var}(\varepsilon_t)$, $\phi_j \geq 0$, $j = 0, 1, \ldots, p$, $\sum_u \phi_u = 1$, $\delta &gt; 0$</td>
</tr>
<tr>
<td>Asymmetric Power–ARCH (APARCH, 1993)</td>
<td>$\sigma_t^2 = \omega + \sum_{u=1}^{r} \alpha_u (</td>
</tr>
<tr>
<td>Quadratic–GARCH (QGARCH, 1993)</td>
<td>$\sigma_t^2 = \phi_0 + \sum_{u=1}^{r} \phi_u y_{t-u} + \sum_{u=1}^{r} \phi_u y_{t-u}^2 + 2 \sum_{u=1}^{r} \sum_{v=u+1}^{r} \phi_{uv} y_{t-u} y_{t-v}$</td>
</tr>
<tr>
<td>Non–linear Asymmetric GARCH (1993)</td>
<td>$\sigma_t^2 = \omega + \sum_{u=1}^{r} \alpha_u (\varepsilon_{t-u} - \gamma_u \sigma_{t-u})^2 + \sum_{v=1}^{s} \gamma_v \sigma_{t-v}^2$</td>
</tr>
</tbody>
</table>

‡ This is the generalized version of the non–linear ARCH, initially proposed by Higgins and Bera (1992).

Likewise, Geweke (1986), Pantula (1986), and Milhoj (1987) studied a linear log–specification for $h$ applied on $\varepsilon_{t-r}$, and $\sigma_{t-r}^2$, called the order $(p, q)$ multiplicative–GARCH, whereas non-linear regimes on $h$ have been explored by Higgins and Bera (1992), Franses and Dick (1996), Sentana (1995), Gokcanm (2000), and Rabiul Alam Beg and Anwar (2010), mostly within the confines of the normal distribution. Table 1.2.1 shows further ARCH–GARCH variants, that is, proposals for $h$ in (1.2.9).

### 1.2.4 Beyond the normality assumption

Extensions of time series models to response distributions in the exponential family also possesses a large literature, and by means of VGLMs, VGLTSMs have been developed for some TS models with non–normal responses. One prominent example is the so–called INGARCH class for the analysis of time series of counts, where VGLTSMs will be shown to confer advantages. The results are presented in Chapter 3, while some application examples are given in Chapter 6. In particular, we focus on discrete response distributions and modelling frameworks for such, where the series is studied through the dynamics governing the conditional mean function, denoted here as $\lambda_t = \mathbb{E}(Y_t|\Phi_{t-1})$. 
1.2. Background

Particularly, Ferland et al. (2006), Fokianos et al. (2009), Fokianos (2011), Fokianos and Fried (2012), Elsaied and Fried (2014), Goncalves et al. (2015) and Ahmad and Francq (2016) have explored the observation–driven order \((r, s)\)–INGARCH model, defined as

\[ Y_t|\Phi_{t-1} \sim F(\lambda_t|\Phi_{t-1}; \Phi_{t-1}), \]

\[ g(\lambda_t|\Phi_{t-1}) = \omega + \alpha^T y_{t-r} + \gamma^T \lambda_{t-s}, \]

with \(\lambda_{t-s} = (\lambda_{t-s}, \ldots, \lambda_{t-1})^T\), and \(g\) a GLM–link. The options, however, for the distributional assumption \(F\) are limited to Poisson, binomial and negative binomial.

Further alternatives are Dunsmuir and Scott (2015) who assembled the residual–driven generalized linear autoregressive moving–average (GLARMA) class or the generalized additive models for location, scale, and shape (GAMLSS), introduced by Rigby and Stasinopoulos (2005b), allowing additional linear predictors, compared to INGARCH–types. Although initially proposed for semiparametric regression analysis, GAMLSS can estimate a range of INGARCH models for time series of counts, as well as incorporate random effects in the analysis. Indeed, Chapter 6 gives some results from VGLTSMs using real data that are to be compared to the GAMLSS and GLARMA modelling frameworks using its companion \(R\) software.

1.2.5 PIT plots

Importantly however, assessing the predictive ability of TS models is often crucial allowing, e.g., to compare and classify a set of efficient forecasting methods, in such a way that a number of calibration methodologies to investigate the statistical consistency between the predictive distribution and the observations have been studied (Dawid, 1984; Diebold et al., 1998; Gneiting et al., 2007)

Currently, this thesis and its software implementation address this subject for time series of counts only. The methodology adopted conforms with Czado et al. (2009), based on non–randomized probability integral transform (PIT) values. Specifically, say \(\{y_t\}\) is an observed TSC and \(\mathcal{P}(y_t) = \mathcal{P}(Y_t \leq y_t|\Phi_{t-1})\) is the (prespecified) cumulative predictive distribution of the true data generating process (DGP). The referred PIT values for \(\{y_t\}\) and \(\mathcal{P}(y_t)\) are defined by Czado et al. (2009) as follows:

\[ F^*(u|\Phi_{t-1}) = \begin{cases} 0, & u \leq \mathcal{P}(y_t - 1|\Phi_{t-1}), \\ \frac{u - \mathcal{P}(y_t - 1|\Phi_{t-1})}{\mathcal{P}(y_t|\Phi_{t-1}) - \mathcal{P}(y_t - 1|\Phi_{t-1})}, & \mathcal{P}(y_t - 1) < u < \mathcal{P}(y_t|\Phi_{t-1}), \\ 1, & u \geq \mathcal{P}(y_t|\Phi_{t-1}), \end{cases} \]  

for a length–fixed sequence \(U = \{u, 0 \leq u \leq 1\}\). If the predictive distribution \(\mathcal{P}(y_t)\) aligns with the true DGP, then the PIT is expected to follow a uniform distribution.
Table 1.2.2. Popular R packages to fit univariate ARMA–GARCH–type models.

<table>
<thead>
<tr>
<th>R package</th>
<th>Version</th>
<th>Published†</th>
<th>Depends on</th>
<th>Underlying optimization or fitting function</th>
</tr>
</thead>
<tbody>
<tr>
<td>arima()</td>
<td>Not a package ‡</td>
<td>–</td>
<td>–</td>
<td>optim().</td>
</tr>
<tr>
<td>FitARMA</td>
<td>1.96</td>
<td>2015–02–19</td>
<td>FitAR, leaps, ltsa, lattice, bestglm</td>
<td>optim() and arima().</td>
</tr>
<tr>
<td>fArma</td>
<td>3042.81</td>
<td>2017–11–18</td>
<td>timeDate, timeSeries, fBasics</td>
<td>arima(), then optim().</td>
</tr>
<tr>
<td>forecast</td>
<td>8.2</td>
<td>2017–09–25</td>
<td>tseries, fracdiff, nnet, colorspace, ggplot2, timeDate</td>
<td>Automatic forecasting of TS, auto.arima() relies on arima().</td>
</tr>
<tr>
<td>tseries</td>
<td>0.10–42</td>
<td>2017–06–22</td>
<td>quadprog, zoo, quantmod</td>
<td>optim(), minimizing the conditional SSE.</td>
</tr>
<tr>
<td>fGarch §∥</td>
<td>3042.83</td>
<td>2017–11–16</td>
<td>timeDate, timeSeries, fBasics</td>
<td>arima(), optimHess(), both based on optim().</td>
</tr>
<tr>
<td>lgarch</td>
<td>0.6–2</td>
<td>2015–09–15</td>
<td>zoo, lattice</td>
<td>optimHess() built upon optim().</td>
</tr>
</tbody>
</table>

† Updated to January 2018.
‡ Function arima() from package stats uses optim() for parameter estimation.
§ fGarch also handles the Student-t and generalized error distributions (GED). (Nelson, 1992).
∥ Packages fGarch and timeSeries developed by the Rmetrics team (Diethelm et al., 2013b); URL: https://www.metrics.org.

To examine this conjecture some accurate representation of the distribution of the PITs, $F_*^t(u|\Phi_{t-1})$, also called PIT plot, is consequently required. The alternative supplied by this work allows us to compare the histogram of the PIT values to the density of a standard uniform random variable (Czado et al., 2009). The histograms are to be constructed upon $B$ equally spaced bins, where the the $j$th bin, $b_j$, has weighted height

$$b_j = \mathcal{F}_*^t(b/B) - \mathcal{F}_*^t((b - 1)/B),$$

(1.2.13)

where $\mathcal{F}_*$ is the PIT mean, given by $\mathcal{F}_*^t(u) = \frac{1}{T} \sum_{t=1}^{T} F_*^t(u|\Phi_{t-1})$.

This method, including the histogram construction, has been implemented in VGAMextra through the function PIT(). Section 4.4.1 presents fuller details about PIT() accompanied with an example on its usage.

1.2.6 Software implementations

This work in its entirety is based on R, and the first version of its implementation is VGAMextra 0.0-1. Many details regarding the multiple methods and functions in VGAMextra that concern this thesis are to be given throughout the subsequent chapters. However, while we aim to provide this information with clarity and transparency,
### Table 1.2.3. Some popular R packages to analyze time series of counts.

<table>
<thead>
<tr>
<th>R package</th>
<th>Version</th>
<th>Published</th>
<th>Depends on</th>
<th>Comments</th>
</tr>
</thead>
</table>
| tscoun... | 1.4.1   | 2017–11–25| ltsa       | ● Quasi–MLE–type estimation methods  
● Allows for inclusion of covariates  
● Links: "identity" and "log"  
● Uses optim() and constrOptim() for ML optimization  
● Limited to Poisson and neg. binomial (NB) |
| glarma    | 1.5–0   | 2017–01–25| MASS       | ● MLE–type, conditional on initial values  
● Restricted to Poisson, binomial and NB  
● Model identifiability issues when the degree of serial dependence is misspecified  
● GLARMAs are residual–driven models |
| gamlss    | 5.0–6   | 2017–12–11| gamlss.dist, nml, parallel, gamlss.data | ● MLE–type, allows p–splines and cubic–splines for the additive terms  
● Function garmaFIT() to fit GARMA models does not guarantee stationarity of the fitted model  
● Unable to cope with stochastic seasonality  
● Fixed high order serial dependence not handled (unless all lower orders included) |
| acp       | 2.1     | 2015–12–04| tseries, quantmod | ● MLE–type estimation  
● Allows covariates but restricted to the Poisson distribution  
● Only manages $\tilde{g} = g =$ identity function in model (2.2.3)  
● Incapable to cope stochastic seasonality |
| INLA      | 17.06.20| 2017–06–20|           | ● Bayesian–type framework  
● Limited to the exponential family  
● Unable to model short–term temporal correlation yet  
● Not coping yet with stochastic seasonality for INGARCH models, as in (2.2.3)  
● Unable to “identify” extraordinary events, i.e., smoother fitted values |
| gcmr      | 1.0.0   | 2017–04–28| betareg, car, geoR, Formula, lmtest, nmle, sandwich, sp | ● Likelihood inference for Gaussian copula marginal regression  
● For count TS, limited to Poisson, NB and binomial responses  
● Models time–varying marginal distributions, not allowing the mean to be influenced by current errors  
● Not suitable for 1–step–ahead predictions |

‡ Not available at CRAN. Package INLA is an R–interface to the INLA–project.  
See [http://www.r-inla.org/](http://www.r-inla.org/)

† Updated to January 2018.

several complementary details will be only mentioned due to space limitations, but full information can be found in the VGAMextra manual.
General software to analyze TS data has proliferated in recent years. To date, the CRAN repository comprises many libraries for TS analysis providing a large range of TS modelling frameworks, including ARMA-type, GARCH–type models, as well as implementations for forecasting from time series models; to mention a few, fArma, forecast, FitARMA, fGarch, or lgarch to fit log–GARCH models. Several of such implementations rely on estimation procedures based on MLE under different distributional assumptions (primarily normal). Moreover, in terms of MLE–optimization, the vast majority of such software predominantly rely on stats::arima() and stats::optim(). Especially, the latter serves as a general–purpose optimizer based on the Nelder–Mead, quasi–Newton, and conjugate–gradient algorithms for constrained optimization, and arima() has been built on the top of this. Nonetheless, optimization via gradient–methods such as the Nelder–Mead or L–BFGS–B algorithms implemented in optim() have been documented as likely to lead to convergence issues, for instance, with approximate gradients, mistakenly reporting an optimum without reaching the solution. Disuse and deprecation has been consequently claimed especially in the presence of genuine and better choices nowadays (see, e.g., Nash (2013) and Nash (2014) for a long discussion on this).

As a reference, Table 1.2.2 lists popular R packages available at CRAN handling ARMA–GARCH models, e.g., fArma, including its dependency on other R sources. Table 1.2.3 does so for TSCs where, gamlss (Rigby and Stasinopoulos, 2005b), glarma (Dunsmuir and Scott, 2015), and tscount (Liboschik et al., 2017) represent a popular subset among the MLE–type estimation platforms.

### 1.3 Vector generalized linear models

VGLMs (Yee, 2015; Yee and Hastie, 2003) provide the engine and central modelling framework in this work. The most important details of this class of models to be used throughout this thesis are now summarized.

Unlike GLMs that are intertwined with the exponential family, VGLMs encompass a much wider class of statistical models with infrastructure to handle multiple linear predictors. Suppose that a $Q$–dimensional $y$ is the observed response. VGLMs are defined in terms of $M$ linear predictors, $\eta = (\eta_1, \ldots, \eta_M)^T$, as a model for which the conditional density of $y$ given a $p$–dimensional vector of explanatories, $x = (x_1, x_2, \ldots, x_p)^T$ with ordinarily $x_1 \equiv 1$ for an intercept, is of the form

$$F(y|x; B) = h(y, \eta_1, \ldots, \eta_M, \phi; x),$$

(1.3.1)

for some known function $h(\cdot)$, with $B = (\beta_1 \beta_2 \cdots \beta_M)$, a $p \times M$ matrix of unknown regression coefficients.
### 1.3. Vector generalized linear models

Table 1.3.1. Some VGLM/VGAM link functions available in VGAMextra. Some are shared with VGAM.

<table>
<thead>
<tr>
<th>Functions</th>
<th>Links (g_j(\theta_j))</th>
<th>Domain of (\theta_j)</th>
<th>Link names</th>
</tr>
</thead>
<tbody>
<tr>
<td>loglink()</td>
<td>(\log \theta_j)</td>
<td>(0, (\infty))</td>
<td>Logarithmic</td>
</tr>
<tr>
<td>clogloglink()</td>
<td>(\log(-\log(1 - \theta_j)))</td>
<td>(0, 1)</td>
<td>Complementary log–log</td>
</tr>
<tr>
<td>logitlink()</td>
<td>(\log \frac{\theta_j}{1 - \theta_j})</td>
<td>(0, 1)</td>
<td>Logit</td>
</tr>
<tr>
<td>logffMeanlink()</td>
<td>(\log(\eta_j) - \text{cloglog}(\eta_j))</td>
<td>(0, 1)</td>
<td>logffMeanlink</td>
</tr>
<tr>
<td>rhobitlink()</td>
<td>(\log\left(\frac{1 + \theta_j}{1 - \theta_j}\right))</td>
<td>((-1, 1))</td>
<td>rhobit</td>
</tr>
</tbody>
</table>

\[\text{‡ This is the VGLM–link for the mean function of the logarithmic distribution (see Section 7.3.1).}\]

In general, the \(\eta_j\) of VGLMs may be applied directly to the parameters, \(\theta_j\), of any distribution, transformed if necessary, as

\[
\eta_j = g_j(\theta) = \eta_j(x) = \beta_j^T x = \sum_{k=1}^{p} \beta_{(j)k} x_k, \quad j = 1, \ldots, M,
\]

where \(g_j\) is a VGLM–parameter link function as in Table 1.3.1 (see Yee, 2015, for further choices) and \(\beta_{(j)k}\) is the \(k\)th element of \(\beta_j\). In matrix form, where \(\beta_{(k)} = (\beta_{(1)k}, \beta_{(2)k}, \ldots, \beta_{(M)k})^T\), \(k = 1, \ldots, p\), one can write

\[
\eta_j(x) = o_i + \beta_j^T x = o_i + \sum_{k=1}^{p} \beta_{(k)j} x_k = B_j^T x.
\]

Sometimes, for some \(j\), it may be required to model \(\eta_j\) as intercept–only, that is, \(\eta_j = \beta_{(j)1}\).

#### 1.3.1 Constraint matrices

VGLMs may accommodate linear relationships between coefficients \(\beta_{(j)p}\) (and hence between parameters) by use of constraint matrices over the linear predictor components. Specifically, denote \(x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T\) the explanatory vector for the \(i\)th observation, and allow a vector of known offsets \(o_i\), then the linear predictors (1.3.3) can be written as

\[
\eta(x_i) = o_i + I_M \beta_{(1)} x_{i1} + \cdots + I_M \beta_{(p)} x_{ip}
\]

\[
= o_i + H_k \beta_{(k)}^* x_{ik} = o_i + \sum_{k=1}^{p} H_k \beta_{(k)}^* x_{ik} = o_i + \sum_{k=1}^{p} \text{diag}(x_{ik1}, x_{ik2}, \ldots, x_{ikM}) H_k \beta_{(k)}^*,
\]
where $H_1, \ldots, H_p$ are known *constraint matrices* of full column–rank and $\beta^{(k)}$ as in (1.3.3). The $\beta^*_k$ are vectors of the form

$$
\beta^*_k = (\beta^*_{(1)k}, \beta^*_{(2)k}, \ldots, \beta^*_{(R_k)k})^T, \quad 1 \leq k \leq p,
$$

(1.3.5)

where $\text{ncol}(H_k) = R_k$, while the $\beta^*_{(j)k}$ are unknown coefficients that are to be estimated. Hence, the matrix $B$ acquires the form (cf. (1.3.1))

$$
B = \left( H_1 \beta^*_{(1)} \mid H_2 \beta^*_{(2)} \mid \cdots \mid H_p \beta^*_{(p)} \right).
$$

(1.3.6)

The matrices $H_k$ can constrain the effect of a covariate over some $\eta_j$ and to have no effect for others. Usually its elements are 0 or 1. With no constraints, then $H_k = I_M$ and $\beta^*_{(k)} = \beta_{(k)}$. By means of VGLMs, constraint matrices play a central role for VGLTSMs and signify new modelling alternatives to popular packages in CRAN for TS analysis, but not available elsewhere. These features allow one to incorporate and conveniently manage parameter constraints over any set of explanatorys, and are accessible through different arguments (such as *zero*) in the corresponding implementations in *VGAMextra*. These arguments will be fully described in Chapter 3 and Chapter 4, while Section 6.2 presents an example where such facilities confers advantages over preceding modelling structures.

### 1.3.2 Reduced–rank VGLMs

A prominent sub–class of VGLMs is the *reduced–rank–VGLM* (RR–VGLM) class which arises by endowing VGLMs with reduced–rank regression (RRR; Izenman, 1975) capabilities. Briefly, partition $x$ into $(x^T_1, x^T_2)^T$ and $B = (B^T_1, B^T_2)^T$ accordingly, with $\dim(x_i) = p_i$, so that $p_1 + p_2 = p$. RR–VGLMs suit situations where $M$ and $p$ are “too” large for the given sample size $n$, by approximating $B_2$ using RRR by the product of two thin matrices $A^T$ and $C$, both of rank $R \ll \min(M, p_2)$, in the form $B_2 = AC^T$. The immediate effect is a significant reduction of the number of regression coefficients to be estimated when $R$ is small. The linear predictor then acquires the following expression:

$$
\eta = B^T_1 x_1 + AC^T x_2.
$$

(1.3.7)

The modelling function for RR–VGLMs is *VGAM::rrvglm()*.

It is also possible to allow some $\eta_j$ to not be subject of RRR by letting $A$ have rows of 0s. This feature is managed by the argument *str0* in *rrvglm()*, where its name stands for *structural zeros*. More details are given in the *rrvglm()* documentation file.
1.3.3 On VGLM estimation

VGLMs are estimated by maximum likelihood performed by iteratively re-weighted least squares (IRLS) using Fisher scoring. The “general” VGLM log-likelihood is given by

$$\ell(\eta) = \sum_{i=1}^{n} w_i \ell_i(\eta_1(x_i), \ldots, \eta_M(x_i)),$$  \hspace{1cm} (1.3.8)

for known fixed positive prior weights $w_i$, and a Newton-like algorithm for maximizing (1.3.8) has the form $\beta^{(a)} = \beta^{(a-1)} + \mathcal{I} (\beta^{(a-1)})^{-1} \mathbf{U} (\beta^{(a-1)})$. The vector $\beta^{(a)}$ is obtained as the solution of the generalized least squares problem $\beta^{(a)} = \arg\min_{\beta} \text{ResSS}$, where the quantity minimized at each IRLS iteration, $a$, is the weighted (or residual) sum of squares, $\text{ResSS} = \sum_{i=1}^{n} w_i \left\{ z_i^{(a-1)} - \eta_i^{(a-1)} (\beta^{(a-1)}) \right\}^T W_i^{(a-1)} \left\{ z_i^{(a-1)} - \eta_i^{(a-1)} (\beta^{(a-1)}) \right\}$, \hspace{1cm} (1.3.9)

Specifically, $\beta^{(a)}$ is the solution of

$$z^{(a-1)} = X_{VLM} \beta^{(a)} + W^{-(a-1)} U^{(a-1)}$$  \hspace{1cm} (1.3.10)

with

$$X_{VLM} = \left( X_1^T, \ldots, X_n^T \right)^T,$$  \hspace{1cm} (1.3.11)

a ‘big’ matrix of order $nM \times Mp$, where $X_i$ is the $M \times (Mp)$ block-matrix $X_i = x_i \otimes I_M$.

For this, the $z_i$ in (1.3.9) are working responses iteratively regressed upon $X_i$, through

$$z_i^{(a-1)} = X_i \beta^{(a-1)} + W_i^{(a)} u_i^{(a-1)} = \eta_i^{(a-1)} + W_i^{(a)} u_i^{(a-1)},$$  \hspace{1cm} (1.3.12)

where the score vector, $u_i$, has $j$th element

$$(u_i)_j = \frac{\partial \ell_i}{\partial \eta_j}. $$  \hspace{1cm} (1.3.13)

The $W_i$'s are the “general” $M \times M$ working weight matrices, with $(j,k)$th element

$$[W_i]_{j,k} = -w_i \mathbb{E} \left( \frac{\partial^2 \ell_i}{\partial \eta_j \partial \eta_k} \right),$$  \hspace{1cm} (1.3.14)

that is, the expected information matrices (EIMs), giving place to Fisher scoring.

At convergence, the estimated variance covariance matrix is

$$\hat{\text{Var}}(\hat{\beta}) = \hat{\phi} \left( X_{VLM}^T W^{(a)} X_{VLM} \right)^{-1},$$  \hspace{1cm} (1.3.15)

with $W = \text{diag}(w_1^{-1} W_1^{-1}, \ldots, w_n^{-1} W_n^{-1})$. 

Further aspects on the VGLM–class are provided in Yee (2015), Section 3.2, such as advantages on using the EIMs instead of the observed information matrices (OIM), as well as a simplified description of the IRLS–Fisher scoring algorithm (Algorithm 3.1) underlying VGLM estimation.

1.4 Aims

Endowed with capabilities of Section 1.3, this thesis proposes the class of VGLTSMs as VGLMs directed towards time series data. This work, however, is restricted to non–seasonal structures for TS analysis (proposed as future work), and based on such, the goals are the following:

1. We wish to investigate how the VGLM log–likelihood (1.3.8) and the working weights (1.3.14) need to be adapted for TS data. This is the subject of Chapter 2. The consequences of such adjustment are reflected by several means including the working matrices $W_i$, the ability of VGLTSMs to model volatility over popular modes for TS, and reduced–rank regression simplifying some well known models for cointegrated time series. This work follows previous successful endeavours of developing VGLMs and its non–parametric version (VGAMs) for other data types, such as categorical data (Yee, 2010), extremes (Yee and Stephenson, 2007), and quantile regression (Yee, 2004).

Chapter 2 will also show that many popular univariate time series models can be derived from VGLTSMs as special cases.

2. The resulting inference framework, including regularity conditions for VGLTSMs, the expressions for the estimated covariance matrix, as well as forecasting methods for VGLTSMs, currently developed at a very early stage, are presented in Chapter 2.

3. Chapter 3 introduces the sub–class of VGLM family functions describing VGLTSMs, abbreviated to “VGLTSMffs”, currently implemented in VGAMextra. Complementary functions such as non–randomized probability integral transform histograms and unit root tests, and its connections with the VGLTSM–framework are surveyed here.

4. Chapter 4 gives ancillary material to Chapter 3 by providing supplementary modelling features of VGLTSMs. Several alternatives based on popular TS models are conveyed by VGLTSMs, and are to be illustrated here. For this, we use examples with simulated data that are compared to other relevant software for TS analysis. R code is included for reproducibility.
5. *Cointegrated time series* are amenable to VGLMs by way of VGLTSMs. This is the subject of Chapter 5, which introduces a VGLTSMff that handles models derived from the order$(K, L)$ error correction models (ECMs; Engle and Granger, 1987) as special case. Our results are compared to other software to analyse CTS. In addition, here we show how the class of RR–VGLMs may help to decrease the number of parameters as $K$ and/or $L$ grow, while preserving the underlying modelling structure.

6. Chapter 6 gives examples of VGLTSMs and VGLTSMff based on artificial and real data. Like in the two previous chapters, our results here are to be compared with alternative R packages, highlighting the advantages conferred over well–known modelling alternatives originally adopted in such.

7. In addition to the primary work on VGLTSMs, new methodology allied to VGLMs was also developed in a few other directions, and is presented in Chapter 7. A central result here is the handling of VGLM–links with $M = 2$ parameters, thus extending the usual 1–parameter specification $\eta_j = g_j(\theta_j)$ to $\eta_j = g_j(\theta_j, \theta_k)$. As an example, we show how this new extension fixes the convergence issues documented towards the estimation of the negative binomial NB-C$2$-2 variant under Fisher scoring (Hilbe, 2011).

8. Some topics derived from this thesis requiring further development using VGLMs and VGLTSMs, e.g., model selection with VGLTSMs, forecasting or seasonal analysis, are discussed in Chapter 8, along with some future directions.

As many different software implementations now exist for fitting various time series models (e.g., a number of libraries in R, Mathworks©, Regression Analysis of Time Series, i.e., the RATS© Econometric Software, or the SAS© Software) we implement a new R package, VGAMextra, to fit VGLTSMs as well as to compare our results to other modelling alternatives for time series analysis.

To ease the reading, most acronyms, notation and glossary are located at the front of this thesis, while some subsidiary material are presented in the Appendices.
Chapter 2

Vector Generalized Linear Time Series Models

2.1 Introduction

This chapter provides the theoretical foundations of this work. It proposes a new sub–class of VGLMs by adding (conditional) past information into the VGLM log–likelihood (cf. (1.3.8)) where the resulting modelling framework is to be shown to accommodate well–known linear variants of observation–driven models for univariate and multiple time series as special cases. Further consequences of such addition, e.g., on the score vector and the EIMs (cf. (1.3.14)), are also detailed.

2.2 Vector generalized linear time series models

Let $\mathbf{x}_t$ be a $p$–dimensional vector allocating both, up to $K$ time–dependent suitable explanatories, denoted $\mathbf{x}_{t,(1)}$, and a $(p – K)$–dimensional vector $\mathbf{x}_{t,(2)}$ of deterministic information (probably from a time series) up to time $t – 1$. This will be denoted and stacked as

$$\mathbf{x}_{t,p} = \mathbf{x}_t = (\mathbf{x}_{t,(1)}^T, \mathbf{x}_{t,(2)}^T)^T = (x_{t,1}, \ldots, x_{t,K-1}, x_{t,K}, x_{t,K+1}, \ldots, x_{t,p})^T,$$  \hspace{1cm} (2.2.1)

where the subscript $t$ is used to emphasize the temporal positions for the covariates.

In the following, suppose a time series, $\mathbf{y}_t = \{y_t; \ t = 1, \ldots, T\}$, is the response observed from the realization of a process, say $\{Y_t, t \in \mathbb{Z}\}$. With $\mathbf{x}_t$ a above, consider the sub-class of VGLMs for which the conditional distribution of $Y_t$, given $\mathbf{x}_t$, has the form:

$$\mathcal{F}(Y_t | \mathbf{x}_t; \mathbf{B}, \Phi_{t-1}) = h_t(\mathbf{y}_t, \eta_{t,1|\phi_{t-1}}, \ldots, \eta_{t,M|\phi_{t-1}}),$$  \hspace{1cm} (2.2.2)
with \( \mathbf{B} = (\beta_1 \beta_2 \cdots \beta_M) \), a \( p \times M \) matrix of coefficients, as in (1.3.1), where \( \phi_t \) denotes the history of the joint process \( \{ (Y_t, x_{t+1}) \} \) up to time \( t \), and \( x_{t,1} \equiv 1 \) for \( t = 1, \ldots, T \), if an intercept is included, as with VGLMs. The coefficient vectors \( \beta_j \) (cf. (1.3.2)) are adapted to match the dimensions of \( x_{t,1} \) and \( x_{t,2} \), as

\[
\mathbf{B}_{j} = \left( \mathbf{B}_{j,K}^T, \mathbf{B}_{j,p-K}^T \right)^T,
\]

with \( \mathbf{B}_{j,K}^T = (\beta_{(j)1}, \ldots, \beta_{(j)K}) \), and \( \mathbf{B}_{j,p-K}^T = (\beta_{(j)K+1}, \ldots, \beta_{(j)p}) \), for \( j = 1, \ldots, M \). The \( j \)th linear predictor acquires the following time–dependent form:

\[
\eta_{t,j|\Phi_{t-1}} = \eta_{t,j} = \mathbf{B}_{j}^T \mathbf{x}_t = \sum_{k=1}^{p} \beta_{(j)k} x_{t,k}, \quad j = 1, \ldots, M.
\]

Likewise, the log–likelihood for models of the form (2.2.2) is given by

\[
\ell(\mathbf{\eta}_t; \mathbf{y}_t; \Phi_{t-1}) = \sum_{t=1}^{T} w_t \ell \left\{ \eta_{t,1}(\mathbf{x}_t), \ldots, \eta_{t,M}(\mathbf{x}_t); \Phi_{t-1} \right\},
\]

for known prior–fixed weights \( w_t \). It transpires that (2.2.5) encompasses a large subclass of VGLMs endowed with capabilities to manage time series data, named the class of VGLTSMs, while (2.2.5) is denominated the VGLTSM–loglikelihood, fully characterizing VGLTSMs.

Similar to VGLMs, VGLTSMs are also capable to accommodate constraints in the parameters. For this, we write (2.2.4) in vector form, with the allowance of known offsets, \( \mathbf{o}_t \), producing (cf. (1.3.4))

\[
\mathbf{\eta}(\mathbf{x}_t) = \mathbf{o}_t + \mathbf{I}_M \beta_{(1)} x_{t,1} + \cdots + \mathbf{I}_M \beta_{(p)} x_{t,p} = \mathbf{o}_t + \sum_{k=1}^{p} \mathbf{H}_k \mathbf{B}_{(k)}^* \mathbf{x}_{t,k},
\]

where \( \beta_{(k)} = (\beta_{(1)k}, \beta_{(2)k}, \cdots, \beta_{(M)k})^T \), \( 1 \leq k \leq p \), or, in vector form (of length \( p \times M \)):

\[
\mathbf{B} = \left( \mathbf{B}_{(1)}^T, \mathbf{B}_{(2)}^T, \ldots, \mathbf{B}_{(p)}^T \right)^T.
\]

As before, the \( \mathbf{H}_k \) are constraint matrices, while the vectors \( \mathbf{B}_{(k)}^* \) acquire a similar form as with (1.3.5), giving place to the corresponding matrix \( \mathbf{B} \) (cf. (1.3.6)) of coefficients \( \beta_{(j)k}^* \) that determine the VGLTSM linear predictors \( \eta_{t,j|\Phi_{t-1}} \) (cf. (2.2.2)), and are to be estimated, \( 1 \leq k \leq p \), \( 1 \leq j \leq M \). Here, we denote \( p_{\text{VLM}} = \sum_{k=1}^{p} \text{ncol}(\mathbf{H}_k) \).

Typically, a univariate time series, say \( \mathbf{y}_t \) as above, with distributional assumption \( f(\mathbf{y}_t|\theta) \) with parameters \( \theta = (\theta_1, \ldots, \theta_M) \), is characterized by its exact log–likelihood:

\[
\ell(\mathbf{\theta}; \mathbf{y}_t) = \sum_{t=1}^{T} w_t \ell(\mathbf{\eta}_t; \mathbf{y}_t; \Phi_{t-1}).
\]
\[
\ell(\theta; y_t, \Phi_{t-1}) = \ell(\theta; y_1, \ldots, y_\kappa, \Phi_{\kappa-1}) + \sum_{t=\kappa+1}^{T} \ell_t(\theta; y_t|\Phi_{t-1}). \tag{2.2.8}
\]

That is, \(\ell(\theta; y_t, \Phi_{t-1})\) is specified by two pieces of information: the marginal log-likelihood, depending on initial values\(^1\) \(y_1, \ldots, y_\kappa\), and the conditional log-likelihood, depending on \(y_t|\Phi_{t-1}\), for \(t > \kappa\). To handle (2.2.8) different procedures have been proposed, e.g., to ignore the first \(\kappa\) observations and then, maximize the conditional part only, or to assume \(y_1, \ldots, y_\kappa\) as independent allowing the marginal log-likelihood to be factorized and hence to maximize the “full” log-likelihood.

However, VGLTSMs are able to handle (2.2.8), and are even more general. Here are a few reasons: Firstly, VGLTSMs allow for convenient parameter transformations via VGLM-link functions, \(\eta_{t,j} = g_j(\theta_j; x_t)\); see Table 1.3.1. Consequently the parameters \(\theta_j, j = 1, \ldots, M\), can be potentially modelled in terms of \(x_t\) through (2.2.4).

For instance, the log-transformed variance or standard deviation between returns from the same market, say \(\sigma^2_{\varepsilon}\), producing \(\eta_{t,2} = \log \sigma^2_{\varepsilon} = \beta^T_2 x_t\). Moreover, VGLTSMs inherit all the modelling features and advantages from VGLMs, such as constraint matrices, to constrain the effect of covariates, say \(x_{t,(2)}\), over specific linear predictors. Also, VGLTSMs handle multiple responses, hence naturally spanning models for multivariate time series, for instance, the class of VARX models, involving covariates, where Section 3.4.1 shows the corresponding implementation assuming the TS—responses are distributed as multivariate normal.

The following sections are devoted to three major classes of observation-driven VGLTSMs defined at present from the VGLM-loglikelihood. There are some segregated, yet relevant, implementations, such as those for VARX models or AR models with ARIMA errors, which are fully addressed in Chapter 3. The first two major classes aforementioned accommodate time series following the classical decomposition

\[
y_{t|\Phi_{t-1}} = \mu_{t|\Phi_{t-1}} + \varepsilon_{t|\Phi_{t-1}}, \tag{2.2.9}
\]

\[\varepsilon_{t|\Phi_{t-1}} \sim F(0, \sigma^2_{\varepsilon|\Phi_{t-1}}),\]

assuming \(\{\mu_{t|\Phi_{t-1}}\}\) is a deterministic sequence of expected returns, and \(\{\varepsilon_{t|\Phi_{t-1}}\}\) a sequence of zero–mean random noise with distributional specification \(F\), conditional on \(\Phi_{t-1}\). At this stage, Gaussian white noise is the choice available in VGAMextra for both classes, however the first steps on further distributional assumptions over the random noise are in progress: Section 3.4.3 gives details on the first implementation in R, derived from VGLTSMs, handling ARMA–like models with Student–\(t\) errors. Extensions to other noise sources is discussed in Chapter 8.

\(^1\)Unfortunately, “initial values” has two meanings in this thesis. The first refers to the marginal log-likelihood term in Eq. (2.2.8). The second refers to the initial values used in the IRLS algorithm which hopefully lead to convergence. Which meaning is being used should be clear from its context.
The third VGLTSM to be proposed allows generalized regression for time series of counts encompassing intervention analysis as well as various other models from the INGARCH class (e.g., Ferland et al., 2006; Fokianos et al., 2009; Fokianos, 2011; Fokianos and Fried, 2012; Elsäied and Fried, 2014; Gonçalves et al., 2015).

The following notation for unknown coefficients, with fixed $u, v, r, s$ positive integers, will be used:

\[ \vartheta = (\vartheta_1, \ldots, \vartheta_u)^T, \quad \phi = (\phi_1, \ldots, \phi_v)^T, \quad \alpha = (\alpha_1, \ldots, \alpha_r)^T, \]
\[ \zeta = (\zeta_1, \ldots, \zeta_s)^T, \quad \gamma = (\gamma_1, \ldots, \gamma_s)^T. \]

The backshift operator, $B$, will also be referred to, as well as $\vartheta(B) = 1 - \vartheta_1 B - \cdots - \vartheta_u B^u$, \hfill (2.2.10)
\[ \phi(B) = 1 - \phi_1 B - \cdots - \phi_v B^v. \]

that is, the AR and MA polynomials of orders $u$ and $v$. Lastly, $\nabla^d = (1 - B)^d$, for integers $d \geq 0$, will symbolize the order–$d$ difference operator.

### 2.2.1 Order–$(u, d, v)$ VGLM–ARIMAX

This VGLTSM is denoted as VGLM–ARIMAX$(u, d, v)$ and can handle weakly non–stationary series in the sense of non–constant mean and non–constant variance. Here, $d$ is the order of differencing. For such, it may impose an order–$(u, v)$ ARMAX structure on the conditional mean model with explanatories $x_{t,(1)}$, which may too be included in the variance model. If required, initial differencing steps up to order $d$ may be applied to the response beforehand. At its simplest, VGLM–ARIMAX$(u, d, v)$ is defined as

\[ \varepsilon_t | \Phi_{t-1} \sim N(0, \sigma^2_{\varepsilon_t|\Phi_{t-1}}), \]
\[ Y_t | \Phi_{t-1} \sim N(\mu_t|\Phi_{t-1}, \sigma^2_{\varepsilon_t|\Phi_{t-1}}), \]
\[ \mu_t|\Phi_{t-1} = \mu_t + \vartheta^T y_{t-u} + \phi^T \varepsilon_{t-v}, \]

where $y_{t-u}^T$ and $\varepsilon_{t-v}^T$ denote the the following lagged values:

\[ y_{t-u} = (y_{t-u}, \ldots, y_{t-1})^T \quad \text{and} \quad \varepsilon_{t-v} = (\varepsilon_{t-v}, \ldots, \varepsilon_{t-1})^T. \] \hfill (2.2.12)

VGLM–ARIMAX handles $M = 2 + u + v$ linear predictors given by

\[ \eta_t = (\mu_t, \log \sigma^2_{\varepsilon_t|\Phi_{t-1}}, \vartheta^T, \phi^T)^T. \] \hfill (2.2.13)
2.2. Vector generalized linear time series models

Table 2.2.1. Linear predictors handled by VGLM–ARIMAX.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Link ((g_j))</th>
<th>(\theta_j)</th>
<th>Linear predictor ((\eta_{j}, j = 1, \ldots, M))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drift (\dagger)</td>
<td>Identity</td>
<td>(\mu_t)</td>
<td>(\eta_{t,1} = g_1(\mu_{t}) = \mu^*)</td>
</tr>
<tr>
<td>Errors variance (\dagger)</td>
<td>Log</td>
<td>(\sigma^2_{\epsilon_{1}</td>
<td>\phi_{t-1}})</td>
</tr>
<tr>
<td>AR coefficients</td>
<td>Identity</td>
<td>(\vartheta_j)</td>
<td>(\eta_{t,j} = g_j(\vartheta_j) = \vartheta_j, \quad j = 3, \ldots, u + 2)</td>
</tr>
<tr>
<td>MA coefficients</td>
<td>Identity</td>
<td>(\phi_j)</td>
<td>(\eta_{t,j} = g_k(\phi_j) = \phi_j, \quad j = u + 3, \ldots, u + v + 2)</td>
</tr>
</tbody>
</table>

\(\dagger\) When no covariates, \(\eta_{t,1} = \eta_{t} = \mu^*\), i.e., intercept–only.
\(\dagger\) \(\beta_{2,K} = (\beta_{(2)1}, \ldots, \beta_{(2)K})^T\), cf. (2.2.3).

Table 2.2.2. Some ARIMAX–like models handled by VGLM–ARIMAX as special cases.

<table>
<thead>
<tr>
<th>Coefficient constraints(\dagger)</th>
<th>Reduced Model(\dagger)</th>
<th>Handle covariates, (\mathbf{x}_{t,(1)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\beta_{(2)l} = 0, \quad l \geq 2;) and (\varphi_{k} = 0, k &gt; 1)</td>
<td>Order((u))–AR,</td>
<td>No</td>
</tr>
<tr>
<td>(\beta_{(2)l} = 0, \quad l \geq 2;) and (\vartheta_{j} = 0, j \geq 1)</td>
<td>Order((v))–MA</td>
<td>No</td>
</tr>
<tr>
<td>(\beta_{(2)l} = 0, \quad l \geq 2;) and (\vartheta_{j} \neq 0, \varphi_{k} = 0)</td>
<td>Order((u, v))–ARMA</td>
<td>No</td>
</tr>
</tbody>
</table>
| \(\beta_{(2)l} = 0, \quad l \geq 2;\) and \(\vartheta_{j} \neq 0, \varphi_{k} 
eq 0\) | Order\((u, v)\)–ARMAX   | Yes, but embedded in the mean model |

\(\dagger\) \(\beta_{(2)l} = 0, l \geq 2\) implies \(g_2(\sigma^2_{\epsilon_{1}|\phi_{t-1}}) = \beta_{(2)1}\), where \(\beta_{(2)1}\) is an intercept. That is, \(\sigma^2_{\epsilon_{1}|\phi_{t-1}}\) is constant.
\(\dagger\) Provided all VGLM–links \(g_{\nu}\) are the identitylink.

These are described in Table 2.2.1. By default, except by \(\eta_{t,2}\), all the \(\eta_{t,j}\)’s are modelled as intercept–only with \(\eta_{t,1}\) acquiring the form \(\eta_{t,1} = \mu_{t} = \mu^*\), where \(\mu^*\) is a scaled mean that is commonly referred to as the drift parameter. When no explanatories are incorporated, \(\eta_{t,2} = \log \sigma^2_{\epsilon_{1}|\phi_{t-1}}\), is in addition, intercept–only thus resembling the typical assumption of constant variance for (weak) stationary ARMA, viz. \(\log \sigma^2_{\epsilon_{1}|\phi_{t-1}} = \beta_{(2)1}\).

Several other modelling alternatives are available for the linear predictors in VGLM–ARIMAX. For instance, in TS analysis it is interesting to explore the effect of other factors, \(\mathbf{x}_{t,(1)}\), over \(\mu_{t}|\phi_{t-1}\), which suggests \(\mu_{t} = \beta_{1,K}^T \mathbf{x}_{t,(1)}\). The default is \(\mu_{t} = \beta_{(1),1} \cdot (1) = \mu^*\), regardless \(\mathbf{x}_{t,(1)}\). Besides, VGLM–ARIMAX has the ability to operate several ARIMAX–like models; Table 2.2.2 shows some options. Note that the VGLM–links, \(g_{\nu}\)’s, must be restricted to identitylink, whilst \(\sigma^2_{\epsilon_{1}|\phi_{t-1}}\) is constrained to be “intercept–only”, thus enforcing stationary conditions for backward compatibility with standard ARIMA and ARIMAX–type models. This and other modelling features and implementation details concerning VGLM–ARIMAX, are described in Chapter 3.
2.2.2 Order–(u, v, r, s) VGLM–ARMAX–GARCH

Denoted by VGLM–ARMAX(u, v–)GARCH(r, s), this VGLTSM involves the modelling of stochastic volatility by imposing an order–(r, s) GARCH–like structure over the conditional variance, \( \sigma^2_{\varepsilon_t|\phi_{t-1}} \), but, further, also admits (suitable) explanatories \( x_{t(1)} \) (cf. (2.2.1)) allowing external factor whose effect over such are to be explored. In addition, an order(u, v)–ARMAX model may be placed over the conditional mean, \( \mu_t|\phi_{t-1} \), where the “innovations” \( \{ \varepsilon_t|\phi_{t-1} \} \), or error terms with respect to the mean process, are a primary series derived from two independent processes: (a) strong white noise (Gaussian at this stage), \( z_t \sim N(0, 1) \), plus (b) a deterministic component, denoted \( \sigma_{\varepsilon_t|\phi_{t-1}} \) which specifies the GARCH structure over this VGLTSM. Concretely, VGLM–ARMAX–GARCH is defined as

\[
\varepsilon_t|\phi_{t-1} = z_t \cdot \sigma_{\varepsilon_t|\phi_{t-1}}, \\
Y_t|\phi_{t-1} \sim N(\mu_t|\phi_{t-1}, \sigma^2_{\varepsilon_t|\phi_{t-1}}), \\
\sigma^2_{\varepsilon_t|\phi_{t-1}} = \omega + \alpha^T G_1(\varepsilon_{t-r}|\phi_{t-1}) + \gamma^T G_2(\sigma_{\varepsilon_{t-s}}|\phi_{t-1}) - \mu_t + \theta^T y_{t-u} + \phi^T \varepsilon_{t-v},
\]

where \( \mu_t = \beta^T_{2K} x_{t(1)} \) by default, while \( \varepsilon_{t-r}|\phi_{t-1} \) and \( \sigma_{\varepsilon_{t-s}}|\phi_{t-1} \) denote the following vectors of lagged values:

\[
\varepsilon_{t-r}|\phi_{t-1} = \left( \varepsilon_{t-r}|\phi_{t-1-(r+1)}, \ldots, \varepsilon_{t-1}|\phi_{t-1} \right)^T, \quad \sigma_{\varepsilon_{t-s}}|\phi_{t-1} = \left( \sigma_{\varepsilon_{t-s}}|\phi_{t-1-(s+1)), \ldots, \sigma_{\varepsilon_{t-1}}|\phi_{t-1} \right)^T.
\]

\( G_1 \) and \( G_2 \) signify real–valued acceptable functions, that do not compromise the positiveness of \( \sigma^2_{\varepsilon_t|\phi_{t-1}} \), and are to be evaluated respectively as

\[
G_1(\varepsilon_{t-r}|\phi_{t-1}) = \left( G_1(\varepsilon_{t-r}|\phi_{t-1-(r+1)}), \ldots, G_1(\varepsilon_{t-1}|\phi_{t-1}) \right)^T, \quad \text{and}
\]

### Table 2.2.3. Linear predictors handled by VGLM–ARMAX–GARCH.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Link ((g_j))</th>
<th>( \theta_j )</th>
<th>Linear predictor ((\eta_{j}, j = 1, \ldots, M))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept (\dagger)</td>
<td>Identity (\mu_t)</td>
<td>(\eta_{t,1} = g_1(\mu_t) = \beta_{1,K}^T x_{t(1)} = \mu^* + \sum_{i=2}^{K} \beta_{(1)i} x_{t,i} )</td>
<td></td>
</tr>
<tr>
<td>Forecast (\dagger)</td>
<td>(\log) (\sigma^2_{\varepsilon_t</td>
<td>\phi_{t-1}})</td>
<td>(\eta_{t,2</td>
</tr>
<tr>
<td>AR coefficients</td>
<td>Identity (\vartheta_j)</td>
<td>(\eta_{t,j} = g_j(\vartheta_j) = \vartheta_j, ; j = 3, \ldots, u + 2)</td>
<td></td>
</tr>
<tr>
<td>MA coefficients</td>
<td>Identity (\phi_j)</td>
<td>(\eta_{t,j} = g_j(\phi_j) = \phi_j, ; j = u + 3, \ldots, u + v + 2)</td>
<td></td>
</tr>
</tbody>
</table>

\( \dagger \eta_{t,1} = \mu^* \) if no explanatories \( x_{t(1)} \) are involved.

\( \dagger \beta_{2,K} = (\beta_{(2)1}, \ldots, \beta_{(2)K})^T \) is a \( K \)–dimensional vector of coefficients (cf. (2.2.3)).
Table 2.2.4. Special cases of conditional variance models handled by VGLM–ARMAX–GARCH.

<table>
<thead>
<tr>
<th>Model ▲</th>
<th>Conditional variance ▲</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear ARCH (LARCH)</td>
<td>( \sigma^2_{\varepsilon_t</td>
</tr>
<tr>
<td>Generalized–ARCH (GARCH)</td>
<td>( \sigma^2_{\varepsilon_t</td>
</tr>
<tr>
<td>Integrated–ARCH (IGARCH)</td>
<td>( \sigma^2_{\varepsilon_t</td>
</tr>
<tr>
<td>Taylor–Schwert</td>
<td>( \sigma_{\varepsilon_t</td>
</tr>
<tr>
<td>Asymmetric–GARCH (AGARCH)</td>
<td>( \sigma^2_{\varepsilon_t</td>
</tr>
<tr>
<td>LogSD–GARCH (Log–GARCH)</td>
<td>( \log \sigma_{\varepsilon_t</td>
</tr>
<tr>
<td>Multiplicative–GARCH (M-GARCH)</td>
<td>( \log \sigma^2_{\varepsilon_t</td>
</tr>
</tbody>
</table>

▲ For backward compatibility, all models require \( \beta_{(2)k} = 0, k = 1, \ldots, K \), that is, no covariates \( x_{t,(1)} \) are admitted.

▲ \( \omega, \alpha^T, \zeta^T, \) and \( \gamma^T \) are vector of coefficients that are to be estimated, while \( \varepsilon_{t-r} \) denotes \( \varepsilon_{t-r|\Phi_{t-1}} \) and \( \sigma_{\varepsilon_{t-s}|\Phi_{t-1}} \) denotes \( \sigma_{\varepsilon_{t-s}|\Phi_{t-1}} \) for simplicity.

\[
G_2(\sigma_{\varepsilon_{t-s}|\Phi_{t-1}}) = \left( G_2(\sigma_{\varepsilon_{t-s}|\Phi_{t-(s+1)}}), \ldots, G_2(\sigma_{\varepsilon_{t-1}|\Phi_{t-1}}) \right)^T.
\]

As with VGLM–ARIMAX, VGLM–ARMAX–GARCH handles \( M = u + v + 2 \) linear predictors, given by

\[
\eta_t = (\mu_t, \log \sigma^2_{\varepsilon_{t-s}|\Phi_{t-1}}, \theta^T, \phi^T)^T.
\]

Note that \( \eta_{t,1} = \mu_t \), and \( \eta_{t,2} = \log \sigma^2_{\varepsilon_{t-s}|\Phi_{t-1}} \) occupy the first two positions. Table 2.2.3 gives the complete list including a short description. On the contrary, however, there are central differences between both modelling frameworks, as follows.

(i) By default VGLM–ARMAX–GARCH handles

\[
\eta_{t,1} = \mu_t = \beta_{(1,K)}^T x_{t,(1)} = \mu^* + \sum_{i=2}^{K} \beta_{(1),i} x_{t,i}
\]

when covariates \( x_{t,(1)} \) are incorporated in the model, unlike VGLM–ARIMA where such are embedded in the forecast variance model and \( \eta_{t,1} = \mu_t \) is intercept–only. Nevertheless, VGLM–ARMAX–GARCH can likewise handle the inclusion of such covariates on its variance model. This facility relies on technical details of the VGLM–ARMAX–GARCH software implementation that will be addressed in Section 3.3.4.

(ii) Further, VGLM–ARMAX–GARCH is more flexible in nature towards condition-
ally modelling $\sigma^2_{\varepsilon_t|\Phi_{t-1}}$, depending on the functions $G_1$ and $G_2$. Some special cases managed by this VGLTSM are shown in Table 2.2.4, but is not restricted to such. Here, $\omega, \alpha^T, \zeta^T$, and $\gamma^T$ are vector of coefficients (cf. (2.2.2)) that are to be estimated.

(iii) If required, the ARMAX–like model imposed over $\mu_{t|\Phi_{t-1}}$ can be disregarded such that pure GARCH–like structures are also handled by VGLM–ARMAX–GARCH. This is illustrated later, in Section 3.3.4. VGLM–ARIMAX, in contrast, fixes the ARMA structure on $\mu_{t|\Phi_{t-1}}$ and this cannot by modified.

It should be noted additionally that VGLM–ARMAX–GARCH manages $x_{t,(2)}$ (cf. (2.2.1)) as the $(r+s)$–dimensional vector

$$x_{t,(2)} = \left(\varepsilon^T_{t-r|\Phi_{t-1}}, \sigma^T_{\varepsilon_{t-r}|\Phi_{t-1}}\right)^T,$$  \hspace{1cm} \text{(2.2.15)}

that is incorporated in $\eta_{t,2} = \log \sigma^2_{\varepsilon_t|\Phi_{t-1}}$, and consequently embedded in the big model matrix $X_{VLM}$ for estimation purposes.

2.2.3 Order($u$, $v$) VGLM–INGARCH with interventions

The third VGLTSM class derived from this work is a general parametric–regression framework for the analysis of TSCs called VGLM–INGARCH of order($u$, $v$), and denoted VGLM–INGARCH($u$, $v$), which allows to model the dynamics of discrete–time data, $\{Y_t; t \in \mathbb{N}\}$, through the process generated by the conditional mean, $\{\mathbb{E}(Y_t|\Phi_{t-1}); t \in \mathbb{N}\}$, in the form $g^{-1}(\mathbb{E}(Y_t|\Phi_{t-1}))$, with $g$ a VGLM–link function.

More precisely, denote

$$\lambda_{t|\Phi_{t-1}} = g^{-1}(\mathbb{E}(Y_t|\Phi_{t-1})),$$

and consider the process $\{\eta_{t|\Phi_{t-1}}; t \in \mathbb{N}\}$ of linear predictors determined by

$$\eta_{t|\Phi_{t-1}} = \beta^T_1 x_t = g(\lambda_{t|\Phi_{t-1}}), \hspace{1cm} t = 1, 2, \ldots.$$

for any set of explanatory, $x_t$, possibly other TSs, where $\Phi_t$ denotes the history of the joint process $\{(Y_t, x_{t+1}, \mathbb{E}(Y_t|\Phi_{t-1})); t \in \mathbb{N}\}$ up to time $t$, and $\beta_1$ are the corresponding coefficients. By specifying a DGP to generate the series $\{\eta_{t|\Phi_{t-1}}; t \in \mathbb{N}\}$, the VGLM–INGARCH class emerges handling one linear predictor, $\lambda_{t|\Phi_{t-1}}$, which combines an INGARCH–like structure (involving $x_t$) with the ability to investigate intervention effects on the location. Intervention analysis has been a subject matter since mid–1970’s, e.g., Box and Tiao (1975) proposed difference equation models attempting
2.2. Vector generalized linear time series models

to collect the intervention dynamics in presence of noise. VGLM–INGARCH allows modelling of up to \( s \) single different types of interventions following the methodology introduced by Fokianos and Fried (2010, 2012), thus incorporating covariates of the form

\[
\delta_i^{t-\tau_i} \mathbf{1} (t \geq \tau_i),
\]

\( i = 1, \ldots, s \), where \( \tau_i \) is the \( i^{th} \) time of occurrence and \( \delta_i \) is a constant (fixed) decay rate in \([0, 1]\), hence, allowing for singular (\( \delta_i = 0 \)), exponential (\( \delta_i \in (0, 1) \)) and permanent (\( \delta_i = 1 \)) effects. However, VGLM–INGARCH is broader still as it admits regression on arbitrary past information and up to order–2 interaction terms over known intervention–exponential effects to investigate its combined impact on the dynamics of the process. The latter structure is similar to the “interaction” effects in standard linear regression, and explores the combined influence on the series of “afterchocks” that may follow one extraordinary event, in presence of the main occurrence, e.g., a large earthquake with subsequent minor events following.

The general form of VGLM–INGARCH(\( u, v \)) with \( s \) interventions, \( \mathbf{\tau} = (\tau_1, \ldots, \tau_s)^T \), at rates \( \mathbf{\delta} = (\delta_1, \ldots, \delta_s)^T \), is given by

\[
Y_t|\phi_{t-1} \sim \mathcal{F}(\lambda_{t|\phi_{t-1}}; \mathbf{x}_t, \Phi_{t-1}),
\]

\[
g(\lambda_{t|\phi_{t-1}}) = \omega + \beta_{1,K}^T \mathbf{x}_{t,(1)} + \mathbf{\vartheta}^T \mathcal{U}(\mathbf{y}_{t,i_k}) + \mathbf{\phi}^T g(\lambda_{t,j_k}) + \mathbf{w}^T \mathbf{\Delta}_{t,\mathbf{\tau},\mathbf{\delta}}.
\]

Here, \( \omega \) is an intercept, \( \mathcal{U} \) is any real–valued function, and \( g \) is any VGLM–link. Also,

\[
\mathbf{y}_{t,i_k} = (y_{t-i_1}, y_{t-i_2}, \ldots, y_{t-i_u})^T,
\]

\[
\mathbf{\lambda}_{t,j_k} = (\lambda_{t-j_1|\phi_{t-j_1}}, \lambda_{t-j_2|\phi_{t-j_2}}, \ldots, \lambda_{t-j_v|\phi_{t-j_v}})^T,
\]

\[
\mathcal{U}^{(i_k)}(\mathbf{y}_{t,i_k}) = (\mathcal{U}(y_{t-i_1}), \ldots, \mathcal{U}(y_{t-i_u}))^T,
\]

\[
g(\mathbf{\lambda}_{t,j_k}) = (g(\lambda_{t-j_1|\phi_{t-j_1}}), \ldots, g(\lambda_{t-j_v|\phi_{t-j_v}}))^T,
\]

for arbitrary sets of positive integers \( \mathcal{U} = \{i_1, i_2, \ldots, i_u\} \), and \( \mathcal{V} = \{j_1, j_2, \ldots, j_v\} \), with \( i_\kappa < i_\bar{k} < \infty \), and \( j_\nu < i_\bar{\nu} < \infty \), for any \( \kappa < \bar{k}, \nu < \bar{\nu} \).

As modelling options for the discrete–response, Table 2.2.5 gives details on the 1–parameter (conditional) VGLM–distributions, \( \mathcal{F} \), supported by VGLM–INGARCH. Additionally, \( \mathbf{\Delta}_{t,\mathbf{\tau},\mathbf{\delta}} \) is an \( s \times s \) symmetric matrix in the matrix–band format (Section 18.3.5 in Yee, 2015) whose \((h, l)\) element at time \( t \) is given by

\[
[\mathbf{\Delta}_{t,\mathbf{\tau},\mathbf{\delta}}]_{h,l} = \begin{cases} 
\delta_h^{t-\tau_h} \mathbf{1} (t \geq \tau_h), & \text{if } h = l, \\
\delta_{\tau_h,\tau_l}, & \text{otherwise},
\end{cases}
\]

(2.2.17)

where

\[
\delta_{\tau_h,\tau_l} = \delta_h^{t-\tau_h} \mathbf{1} (t \geq \tau_h) \cdot \delta_l^{t-\tau_l} \mathbf{1} (t \geq \tau_l).
\]

(2.2.18)
Table 2.2.5. One–parameter VGLM–distributions $\mathcal{F}$ supported by VGLM–INGARCH.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>$f(y; \lambda)$,‡</th>
<th>Comment§</th>
</tr>
</thead>
</table>
| Poisson               | $\frac{\lambda^y \exp(-\lambda)}{y!}$ | • Support: $y \in \{0, 1, 2, \ldots\}$.  
• Mean $\lambda > 0$.  
• Models by default $\loglink(\lambda)$.  
• Dispersion parameter set to 1.  
• Choices handling over–dispersion to be included at a next stage, such as quasi–Poisson. |
| Negative Binomial     | $C^y_{y+k-1} \left(\frac{\lambda}{\lambda + k}\right)^y \left(\frac{k}{k + \lambda}\right)^k$ | • Support: $y \in \{0, 1, 2, \ldots\}$.  
• $\lambda > 0$, is the mean.  
• Models by default $\loglink(\lambda)$.  
• $k > 0$ is an index parameter which can also estimated by VGLM–INGARCH. |
| Logarithmic (log–series) | $-\frac{\lambda^y}{y \cdot \log(1 - \lambda)}$ | • Support: $y \in \{1, 2, \ldots\}$.  
• $0 < \lambda < 1$, is a shape parameter  
• Models by default $\logitlink(\lambda) - \clogloglink(\lambda)$. |
| Yule–Simon            | $y \cdot \text{Be}(y, \lambda + 1)$ | • Support: $y \in \{1, 2, \ldots\}$.  
• $\lambda > 0$, is a shape parameter.  
• Models by default $-\loglink(1 - \lambda^{-1}), \lambda > 1$. |

† The parameter is $\lambda$.
‡ Additional parametrizations of $f(y; \lambda)$ may be available. The ones implemented in the VGLTSM framework are listed here.
§ Further VGLM–links choices for $\lambda$ are available. See Yee (2015), Table 1.2.

Further,

$$w = (w_1, \ldots, w_s, w_{s+1}, \ldots, w_{M^*})^T$$  \hspace{1cm} (2.2.19)

is the corresponding $M^*$–dimensional vector of (intervention) effect sizes. At present, VGLM–INGARCH supports interactions with exponential effects only, i.e., $\delta_h \in (0, 1)$. Hence, for $s < k \leq M^*$, one has

$$w_k = 0, \text{ if } \delta_h \in \{0, 1\} \text{ or } \delta_t \in \{0, 1\}.\$$

VGLM–INGARCH handles one linear predictor, $\lambda_t\phi_{t-1}$, spanning several popular models for time series of counts presented in Table 2.2.6. The majority are variants of (2.2.16), but unable to handle covariates $x_t$ or interactions, and limited to some discrete distributions in the 1–parameter GLM–family. Coefficients $\omega, \vartheta^T, \phi^T$, and $\omega^T$
### Table 2.2.6. INGARCH–type models for time series of counts supported by VGLM–INGARCH: Special cases.

| Model                           | Parameter Constraints | Reduced Model for $\lambda_t|\omega, \phi_{t-1} = \lambda_t$ |
|---------------------------------|-----------------------|--------------------------------|
| Order($u$)–INARCH               |                       | $\lambda_t = \omega + \theta^T y_t,i_n$ |
| (Fokianos et al. (2009), or    | $\phi_v = 0, v \geq 1$|                                |
| Elsaied and Fried (2014))       | $\beta_{(1)l} = 0, l \geq 1$|                                |
|                                 | $w_j = 0, j = 1, \ldots, M^*$|                                |
|                                 | $U = \{1, 2, \ldots, u\}$|                                |
| Order($u$, $v$)–INGARCH         | $\beta_{(1)l} = 0, l \geq 1$| $\lambda_t = \omega + \theta^T y_t,i_n + \phi^T \lambda_{t,j}$ |
| (e.g., Ferland et al. (2006),   | $w_j = 0, j = 1, \ldots, M^*$|                                |
| Fokianos et al. (2009))         | $U = \{1, 2, \ldots, u\}$|                                |
|                                 | $V = \{1, 2, \ldots, v\}$|                                |
| Order($u$, $v$) Log–linear      | $\beta_{(1)l} = 0, l \geq 1$| $\log \lambda_t = \omega + \theta^T \log(y_t,i_n + 1) + \phi^T \log \lambda_{t,j}$ |
| INGARCH                         | $w_j = 0, j = 1, \ldots, M^*$|                                |
| (e.g., Fokianos and             | $U = \{1, 2, \ldots, u\}$|                                |
| Tjostheim (2011))               | $V = \{1, 2, \ldots, v\}$|                                |
| Order($u$, $v$)–INGARCH         | $w_j = 0, j = 1, \ldots, M^*$| $g(\lambda_t) = \omega + \beta_{1,h}^T x_{t,(1)} + \theta^T \tilde{g}(y_t,i_n) + \phi^T g(\lambda_{t,j})$ |
| with covariates, $x_t$          | $g$ is a GLM–link     |                                |
| (Fokianos and Fried (2010),     | $\tilde{g}: N_0 \rightarrow \mathbb{R}$|                                |
| Fokianos and Fried (2012))      |                                      |                                |
| Order($u$, $v$)–INGARCH$^*$     | $\Delta_{t,\tau,\delta} = \text{diag} \{\delta_{i}^{1-\tau}\}$| $g(\lambda_t) = \omega + \beta_{1,h}^T x_{t,(1)} + \theta^T \tilde{g}(y_t,i_n) + \phi^T g(\lambda_{t,j}) + w^T \Delta_{t,\tau,\delta}$ |
| with interventions (No interactions) | $w_j = 0, j > s$           |                                |
| (Fokianos and Fried (2010),     | $g$ is a GLM–link     |                                |
| Fokianos and Fried (2012))      | $\tilde{g}: N_0 \rightarrow \mathbb{R}$|                                |

$^\dagger$ All models are restricted to discrete distributions in the GLM–family, specifically, binomial, Poisson or negative binomial.

$^\ddagger$ Constraints $U = \{1, 2, \ldots, u\}$, and $V = \{1, 2, \ldots, v\}$ indicate that only consecutive, rather than arbitrary, past information is handled.

$^\S$ No interaction terms for interventions admitted, hence $\Delta_{t,\tau,\delta} = \text{diag} \{\delta_{1}^{1-\tau}, \delta_{2}^{1-\tau}, \ldots, \delta_{s}^{1-\tau}\}$

shown in Table 2.2.6 are incorporated in the matrix $\mathbf{B}$ (cf. (2.2.2)) and are to be estimated.

It should be mentioned that the VGLTSM–classes surveyed in Sections 2.2.1–2.2.3 possess a software implementation in R as well as other modelling abilities that, due to its technical nature, are to be addressed in Chapter 3.
2.3 Estimation

As a class derived from VGLMs, VGLTSMs can be estimated by IRLS using Fisher scoring following (1.3.9)–(1.3.12). To maximize the likelihood, we apply the Newton-like algorithm:

$$\beta(a) = \beta(a-1) + \mathcal{I}_t (\beta(a-1))^{-1} U_t (\beta(a-1))^T,$$

(2.3.1)

where $\mathcal{I}_t$ and $U_t$ denote the VGLTSM–EIMs and the VGLTSM–score vector, respectively, surveyed in Section 2.3.1. Then the quantity minimized at the $a$th IRLS–iteration is

$$\text{ResSS}_T = \sum_{t=1}^{T} w_t \left\{ z_t(a-1) - \eta_t(a-1) (\beta(a-1)) \right\}^T W_t(a-1) \left\{ z_t(a-1) - \eta_t(a-1) (\beta(a-1)) \right\},$$

(2.3.2)

where $\beta(a) = \arg\min_\beta \text{ResSS}_T$, and $\eta_t = (\eta_t,1\ldots,\eta_{t,M})^T$, with $\eta_{t,j}$ as in (2.2.4), $j = 1,2\ldots,M$. The $z_t$ are the time–dependent working responses regressed upon $X_t = x_{t,p} \otimes I_M$ as in (1.3.12), i.e.,

$$z_t = X_t \beta + W_t^{-1} u_t = \eta_t + W_t^{-1} u_t,$$

(2.3.3)

supressing the superscript $(a-1)$ for clarity.

Note that (2.3.3) indeed requires the score vector $u_t$ (cf. (1.3.13)) as well as working weight matrices $W_t = W_{t|\Phi_t^{-1}}$ (cf. (1.3.14)) for every VGLTSM–class. This subject is to be delineated in the following section, in addition to some important chain rule formulas with regards to VGLM–links in order to make Fisher scoring operational.

2.3.1 The score vector and the EIMs

From Section 2.2, VGLTSMs have log–likelihood $\ell(\eta; y_t, \Phi_{t-1}) = \ell_{\Phi_{t-1}}$ determined by $M$ linear predictors $\eta_{t,j} = g_j(\theta_j; \Phi_{t-1})$ involving one–parameter $\theta_j$ at the time$^2$.

To manage such functional dependency between $\eta_{t,j}$ and $\theta_j$ the following chain–rule formulas are required at each IRLS–iteration:

$$\frac{\partial \ell_{\Phi_{t-1}}}{\partial \eta_{t,j}} = \frac{\partial \ell_{\Phi_{t-1}}}{\partial \theta_{t,j}} \frac{\partial \theta_{t,j}}{\partial \eta_{t,j}},$$

(2.3.4)

$$-\mathbb{E} \left[ \frac{\partial^2 \ell_{\Phi_{t-1}}}{\partial \eta_{t,j}^2} \right] = -\mathbb{E} \left[ \frac{\partial^2 \ell_{\Phi_{t-1}}}{\partial \theta_{t,j}^2} \left( \frac{\partial \theta_{t,j}}{\partial \eta_{t,j}} \right)^2 \right],$$

$$-\mathbb{E} \left[ \frac{\partial^2 \ell_{\Phi_{t-1}}}{\partial \eta_{t,j} \partial \eta_{t,k}} \right] = -\mathbb{E} \left[ \frac{\partial^2 \ell_{\Phi_{t-1}}}{\partial \theta_{t,j} \partial \theta_{t,k}} \left( \frac{\partial \theta_{t,j}}{\partial \eta_{t,j}} \right) \left( \frac{\partial \theta_{t,k}}{\partial \eta_{t,k}} \right) \right].$$

$^2$Within the VGLM/VGAM framework, $\eta_j$ may in some cases involve more than one parameter. See, e.g., the VGLM–family function negbinomial() from in VGAM
With an abuse of notation, we denote \( \theta_{t,j} = g_j^{-1}(\eta_j; \Phi_{t-1}) \) to emphasize the temporal dependence of \( g_j^{-1}(\cdot) \), viz. the inverse of the VGLM–link \( g_j \), yielding

\[
\frac{\partial \theta_{t,j}}{\partial \eta_j} = \frac{\partial g_j^{-1}}{\partial \theta_j}, \quad j = 1, 2, \ldots, M.
\]  

(2.3.5)

Regarding VGLM–ARIMAX and VGLM–ARMAX–GARCH, both VGLTSMs handle time series conforming with the classical decomposition (2.2.9), where the random innovation are assumed as zero–mean Gaussian processes. Consequently, the log–likelihood for both modelling frameworks in terms of \( \theta \) is then given by

\[
\ell(\theta; y_t, \Phi_{t-1}) = -\frac{T}{2} \log 2\pi - \frac{1}{2} \log |V_t(\theta)| - \frac{1}{2} [y_t - \mu_{\theta_{t-1}}(\theta)]^T V_t^{-1}(\theta) [y_t - \mu_{\theta_{t-1}}(\theta)],
\]

where \( V_t \) is a Toeplitz matrix with elements

\[
[V_t]_{i,j} = \mathbb{E}(y_{t+i-j}, y_t) = \rho_{t,i-j}(\theta), \quad 1 \leq i, j \leq T.
\]

The sequences \( \{\mu_{\theta_{t-1}}(\theta)\} \) and \( \{\rho_{t,i-j}(\theta)\} \) are functions of the \( M \) parameters \( \theta = (\theta_1, \ldots, \theta_M)^T \) pre–specified upon the model in turn.

Differentiating (2.3.6) with respect to \( \theta_j \), and simplified notation \( \ell_{\theta_{t-1}} = \ell(\theta; y_t, \Phi_{t-1}) \), gives the score vector \( \frac{\partial \ell_{\theta_{t-1}}}{\partial \theta_j} =: \)

\[
-\frac{1}{2} \text{trace} \left\{ V_t^{-1}(\theta) \frac{\partial V_t(\theta)}{\partial \theta_j} \right\} + \frac{1}{2} [y_t - \mu_{\theta_{t-1}}(\theta)]^T V_t^{-1}(\theta) \frac{\partial V_t(\theta)}{\partial \theta_j}.
\]

(2.3.7)

Further, the exact EIMs for both VGLTSMs with respect to \( \theta \), whose \( (j, k) \) elements will be denoted as \( [\mathcal{I}_t(\theta)]_{j,k} = -\mathbb{E} \left[ \frac{\partial^2 \ell_{\theta_{t-1}}}{\partial \theta_j \partial \theta_k} \right] \), can be computed directly from (2.3.7) for \( t = 1, \ldots, T \), conforming with Porat and Friedlander (1986). Specifically, the \( (j, k) \) entry for \( t = 1, \ldots, T \), is given by

\[
[\mathcal{I}_t(\theta)]_{j,k} = \frac{1}{2} \text{trace} \left\{ V_t^{-1}(\theta) \frac{\partial V_t(\theta)}{\partial \theta_j} V_t^{-1}(\theta) \frac{\partial V_t(\theta)}{\partial \theta_k} \right\} + \left[ \frac{\partial \mu_{\theta_{t-1}}(\theta)}{\partial \theta_j} \right] V_t^{-1}(\theta) \left[ \frac{\partial \mu_{\theta_{t-1}}(\theta)}{\partial \theta_k} \right].
\]

(2.3.8)
Table 2.3.1. Score vector and EIMs of the statistical models supported by VGLM–INGARCH.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Score vector ((\ell(\theta; y_t, \Phi_{t-1})))</th>
<th>EIMs (([I_T(\theta)]_{j,k}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson</td>
<td>(\frac{y_t - \lambda_{t</td>
<td>\Phi_{t-1}}}{\lambda_{t</td>
</tr>
<tr>
<td>Negative binomial†</td>
<td>(\frac{y_t}{\lambda_{t</td>
<td>\Phi_{t-1}}} - \frac{1 + y_t/k}{1 + \lambda_{t</td>
</tr>
<tr>
<td>Logarithmic‡</td>
<td>(-a \left(1 - \lambda_{t</td>
<td>\Phi_{t-1}}\right) + \frac{y_t}{\lambda_{t</td>
</tr>
<tr>
<td>Yule–Simon§</td>
<td>(\frac{1}{\lambda_{t</td>
<td>\Phi_{t-1}}} + \Psi(1 + \lambda_{t</td>
</tr>
</tbody>
</table>

† \(k\) is an index parameter. See \(R\) documentation of the `negbinomial()` family function in `VGAM`.
‡ The EIMs are approximated by simulation. See \(R\) documentation of the `yulesimon()` family function in `VGAM`. Also \(a = -1/\log(1 - \lambda_{t|\Phi_{t-1}})\).
§ \(\Psi\) is the digamma function.

Alternatively, approximate expressions for the EIMs are also available that hold for sufficiently large number of observations under stationary conditions. Specifically, for VGLM–ARIMAX and VGLM–ARMAX–GARCH, it is approximately given by

\[
[I(\theta)]_{j,k} \approx \frac{T}{\sigma^2_e} \begin{pmatrix}
(2\sigma^2_e)^{-1} & 0^T & 0^T \\
\gamma_0 & \cdots & \gamma_{u-1} & \gamma_{v}^{e,y} & \cdots & \gamma_{v}^{e,y} \\
0 & \vdots & \vdots & \vdots & \vdots & \vdots \\
\gamma_{u-1} & \cdots & \gamma_0 & \gamma_{v}^{e,y} & \cdots & \gamma_{v}^{e,y} \\
\gamma_{v}^{e,y} & \cdots & \gamma_{v}^{e,y} & 1 & \cdots & \gamma_{v}^{e,y} \\
0 & \vdots & \vdots & \vdots & \vdots & \vdots \\
\gamma_{v}^{e,y} & \cdots & \gamma_{v}^{e,y} & \gamma_{v-1}^{e} & \cdots & 1
\end{pmatrix}. \quad (2.3.9)
\]

Here, \(\gamma_{i-j}\) and \(\gamma_{k-l}^{e}\) denote the usual covariances \(\mathbb{E}[(y_{t-i} - \mu) \cdot (y_{t-j} - \mu)]\), and \(\mathbb{E}[(\varepsilon_{t-k} \cdot \varepsilon_{t-l})]\), respectively, while \(\gamma_{u,v}^{e,y}\) are the cross-covariances \(\mathbb{E}[(y_{t-u} - \mu) \cdot \varepsilon_{t-v}]\), where \(\mu = \mathbb{E}(Y_t)\), the unconditional mean of the process. For VGLM–INGARCH, Table 2.3.1 presents the score vector and the EIMs with respect to \(\theta\) involved at IRLS–Fisher scoring according to the distributional assumption for the response (Table 2.2.5).

Therefore, the score vectors and EIMs in terms of the linear predictors, \(\eta_t\), given by (2.3.4), follow from (2.3.5), (2.3.7), (2.3.8), and, optionally, from (2.3.9). The VGLTSM–working weight matrices (cf. (1.3.14)) are then specified by
\[ [W_t(\Phi_{t-1})]_{j,k} = -w_t [I_T(\eta)]_{j,k} = -w_t \mathbb{E} \left[ \frac{\partial^2 \Phi_{t-1}}{\partial \eta_j \partial \eta_k} \right], \quad t = 1, 2, \ldots, T. \quad (2.3.10) \]

All VGLTSM family functions in VGAMextra manage both choices for the EIMs, that is, “exact” and “approximate”, except for VGLM–INGARCH which handles the latter only. Directions on VGLTSM arguments management are given in Chapter 3.

### 2.3.2 Inference

Inference for vector generalized linear time series models is largely covered by MLE theory relevant to the VGLM class. Let

\[ \beta^* = (\beta^*_T(1), \ldots, \beta^*_T(p))^T \quad (2.3.11) \]

be the large vector allocating the \( R = M \times p \) regression coefficients to be estimated, with \( \beta^*_T(k) \) as in (2.2.6), \( k = 1, \ldots, p \). It has been particularly emphasized that the linear predictors \( \eta_t \) are indeed linear functions of the coefficients \( \beta^* \), such that we will use \( I_t(\eta_t) := I_t(\beta^*) \), without loss of generality.

Firstly, the usual regularity conditions, summarized in Section A.1.2.2 of Yee (2015), must be satisfied for Fisher scoring to work adequately. Under such conditions, MLEs provide an inference platform to VGLMs, including confidence intervals (CIs) and hypothesis testing, based on the “asymptotic normality” of MLE estimators constructed from independent observations. VGLTSMs, however, must meet additional conditions to allow correlated data, as well as normality still governing asymptotic distributions, as follows:

1. The process \( \{(x_t, y_t)\} \) has a joint stationary distribution. That is, it obeys a strong/weak law of large numbers with almost sure/probability limit

\[ \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \mathbb{E}(x_t^T \cdot y_t) < \infty, \]

for \( t = 1, 2, \ldots, T \). Note that \( x_t \), and \( y_t \) are the \( t^{th} \) observations.

2. The process \( \{I^{-1/2}_\infty(x_t, \varepsilon_t)\} \) is subject to the central limit theorem, where

\[ I_\infty = \lim_{T \to \infty} \text{Var} \left( \frac{1}{\sqrt{T}} \sum_{t=1}^{T} x_t^T \cdot \varepsilon_t \right) \]
is positive–definite, and $\varepsilon_t$ are the working residuals, $z_t^{(a)} - \eta_t^{(a)}$, at convergence (viz. (1.3.9)). In addition, $E(\varepsilon_t \cdot x_t) = 0$ for all $t$.

Under the aforementioned conditions, an approximate normal–theory $100(1 - \alpha)\%$ CI for coefficients $\beta^*$ towards the true value $\beta^*_R$ is the ellipsoid

$$
(\hat{\beta}^* - \beta^*_R)^T \mathcal{I}_t(\hat{\beta}^*) (\hat{\beta}^* - \beta^*_R) \leq \chi^2_{1 - \alpha}(R).
$$

(2.3.12)

In particular, for the $i^{th}$ coefficient, $\beta^*_i$, an approximate $100(1 - \alpha)\%$ CI is given by

$$
\hat{\beta}^*_i \pm z(\alpha/2) \text{SE}(\hat{\beta}^*_i),
$$

(2.3.13)

where SEs derive from the square root of the diagonal elements of the estimated variance–covariance matrix, which at Fisher–scoring convergence is given by

$$
\hat{\text{Var}}(\hat{\beta}^*) = \hat{\Phi} (X_{\text{VLM},T} W_{T|\phi_{t-1}}^{-1} X_{\text{VLM},T})^{-1}.
$$

(2.3.14)

Here, $W_{T|\phi_{t-1}}^{-1}$ is determined by the working weight matrices $W_{t|\phi_{t-1}}^{-1}$ in the form

$$
W_{T|\phi_{t-1}}^{-1} = \text{Var}(\varepsilon_t) = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1}).
$$

With trivial constraints, the big model matrix $X_{\text{VLM},T}$ is constructed as

$$
X_{\text{VLM},T} \equiv X_{\text{VLM}} \equiv X_{\text{LM},T} \otimes I_M,
$$

(2.3.15)

with $X_{\text{LM},T} = (x_1, x_2, \ldots, x_T)^T$. Here, $x_t$ is the explanatory vector for the $t^{th}$ observation, storing the covariates $x_{t,(1)}$ and $x_{t,(2)}$, as in (2.2.1). The scaling parameters, $\Phi$, are set to unity.

For VGLM–ARIMAX($u$, $d$, $v$), for example, $R = u + v + 2$ is the number of coefficients arranged in $\beta^*$, since the forecast variance, $\sigma^2_{\varepsilon_t|\phi_{t-1}}$, is modelled as intercept–only by default, else, this number may increase up to $R = 1 + u + v + p$. Under above conditions, the asymptotic distribution at any case is given by

$$
T^{1/2}(\hat{\beta}^* - \beta^*_R) \overset{p}{\sim} N_R(0, \mathcal{I}_t^{-1}(\beta^*_R)).
$$

Here $\mathcal{I}_t$ represents either the exact (viz. (2.3.8)) or the approximate (viz. (2.3.9)) EIM.

VGLM–ARMAX($u$, $v$)–GARCH($r$, $s$) handles $M = u + v + 2$ linear predictors although the number of coefficients to be estimated may be up to $R = M + r + 2s$, while VGLM–INGARCH controls up to $R = 1 + u + v + s$ coefficients, where $s$ denotes the number of interventions.
2.4 Forecasting

Only the very early stages of forecasting from VGLTSMs have been developed, involving solely intercept-only models from the sub-class VGLM–ARIMAX\((p, 0, q)\). Its implementation in \texttt{R}, a preliminary \texttt{forecast()} method for VGLM–ARIMAX\((p, 0, q)\), is currently being tested, and is planned to be included at later versions of \texttt{VGAMextra}.

In this section we particularly give the basics of the methodology adopted for this (excluding its implementation in \texttt{R}), which is based on the minimum mean square error criteria for ARMA models largely described in Wei (2006, Chapter 5).

Consider the case \(d = 0\), i.e., VGLM–ARIMAX\((u, 0, v)\) as in \((2.2.11)\), and its generic form in terms of the AR, and MA polynomials:

\[
\vartheta(B)Y_t = \phi(B)\varepsilon_t,
\]

which has a moving average representation under stationary conditions in the form

\[
Y_t = \Psi(B)\varepsilon_t \quad \text{with} \quad \Psi(B) = \frac{\phi(B)}{\vartheta(B)} = 1 + \sum_{j=1}^{\infty} \Psi_j B^j.
\]

Denoting \(Y_{T+l}\) an \(l\)-step ahead, any forecast \(\hat{Y}_{T+l}\) can be expressed as \((2.4.2)\) upon information \(\Phi_{T+l}\), in the form \(\hat{Y}_{T+l} = \sum_{i=0}^{\infty} \Psi_{l+i}^* \varepsilon_{T+l-i}\). The mean square error of the forecast can be expressed in the form

\[
E(Y_{T+l} - \hat{Y}_{T+l})^2 = \sigma^2 \left[ \sum_{i=0}^{l-1} \Psi_i^2 + \sum_{j=0}^{\infty} (\Psi_{l+j} - \Psi_{l+j}^*) \right],
\]

which reaches its minimum when \(\Psi_{l+j} = \Psi_{l+j}^*, \ j = 0, 1, \ldots\), yielding

\[
\hat{Y}_{T+l} = \sum_{i=0}^{\infty} \Psi_{l+i} \varepsilon_{T+l-i} = E(Y_{T+l}|Y_{T+l-1}, Y_{T+l-2}, \ldots),
\]

with variance \(\text{Var}(\hat{Y}_{T+l}) = \sigma^2 \sum_{i=0}^{l-1} \Phi_i^2\).

Forecasting of \(Y_{T+l}\) from VGLM–ARIMAX\((u, 0, v)\) is addressed through its conditional expectation \((2.4.3)\) with (conditional) variance as above. When \(d \neq 0\), the optimal forecast of \(Y_{T+l}\) is also given by \((2.4.3)\) under mild regularity conditions. For the remaining cases, e.g., \(d \neq 0\), and the VGLM–INGARCH class, forecasting methods are planned to be incorporated over time.
2.5 Summary

This chapter gives details on the fundamentals of VGLTSMs focussing on the three major VGLTSM–classes defined at present, viz. VGLM-ARIMA\((u, d, v)\), VGLM–ARMAX\((u, v)\)–GARCH\((r, s)\), and VGLM–INGARCH\((u, v)\). VGLTSMs arise as a sub–class of VGLMs by constraining the VGLM log–likelihood to time–dependent, deterministic information and are likewise estimated by MLE using Fisher scoring, which has been adapted to accommodate VGLTSMs. The essential requirements are the score vectors and the VGLM–weight working matrices, that have been derived accordingly and presented in Section 2.3.

Specially, forecasting methods for VGLTSMs are still under development, although some particulars about this have as well been surveyed. Fuller details on the VGLTSM implementation in \textit{R} as well as on the range of modelling features conveyed are given successive chapters.
Chapter 3

New VGLTSM Family Functions

3.1 Introduction

The aim of this chapter is two–fold: (i) to describe implementation details of VGAMextra, the software implementation of the methodology given in the preceding chapter, and (ii) to provide skeletal aspects on the new VGLTSM family functions (VGLTSMff) currently available in VGAMextra, as well as illustrations of their usage. As a central component, this chapter also gives details on miscellaneous functions in VGAMextra that make VGLTSMffs functional.

VGAMextra requires VGAM and, consequently, any description of VGAMextra is strongly tied to VGAM. Currently, VGAMextra is on CRAN on its version 0.0-1. Note that some of the software details given here are subject to change.

3.2 On VGAMextra and VGAM

VGAM is a software implementation in R of about six major classes of statistical models for regression analysis, including VGLMs and VGAMs as its central engine. The book Yee (2015) gives details on the VGLM/VGAM framework as well as on the major sub–classes, such as reduced–rank VGLMs (RR–VGLMS), and quadratic RR–VGLMs (QRR–VGLMs). VGLMs/VGAMs have shown to confer advantages to ‘ordinary’ regression modelling hindered by a lack of common framework, and has helped to unify many models in areas such as quantile regression and extremes. VGAM operates using the S4–OOP system, while the VGLM fitting process, based on IRLS–Fisher scoring, is chiefly embodied by the modelling function vglm(). Internally, vglm() calls vglm.fit(), which actually performs the computations of the IRLS algorithm, returning an R–object of class "vglm" for which many S4–methods have been implemented, e.g., summary() or fitted(). A simplified description of the underlying IRLS–algorithm implemented in vglm.fit() is provided in Yee (2015, Section 3.2.1).
Table 3.2.1. Some relevant arguments of \texttt{vglm()}.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>A symbolic description of the model to be fit.</td>
<td>Usually, e.g., ( y \sim 1 + x_2 + x_3 ) for covariates ( x_2, x_3 ).</td>
</tr>
<tr>
<td>family</td>
<td>An \texttt{R} function of class &quot;\texttt{vglmff}&quot; describing what statistical model is to be fitted</td>
<td>VGLTSMffs (Table 3.3.1) are entered through this argument. The name ‘family’ is used loosely to help \texttt{glm()} users.</td>
</tr>
<tr>
<td>data</td>
<td>An optional data frame with the variables in \texttt{formula}.</td>
<td>By default, the variables are taken from the environment where \texttt{vglm()} is called</td>
</tr>
<tr>
<td>weights</td>
<td>An optional vector or matrix of prior, fixed positive weights.</td>
<td></td>
</tr>
<tr>
<td>etastart or</td>
<td>(Optional) Starting values for the linear predictors or fitted values.</td>
<td>Some family functions do not make use of these arguments.</td>
</tr>
<tr>
<td>mustart</td>
<td></td>
<td></td>
</tr>
<tr>
<td>constraints</td>
<td>An optional list of constraint matrices, ( H_k ) (viz. (1.3.4)).</td>
<td>Constrain the effect of covariates on the coefficients ( \beta^* ).</td>
</tr>
<tr>
<td>control</td>
<td>A list of parameters for controlling the fitting process</td>
<td></td>
</tr>
<tr>
<td>offset</td>
<td>A vector or ( M )-column matrix of offset values (viz. (1.3.4)).</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Further arguments passed into \texttt{vglm.control()}.</td>
<td>See Yee (2015, Section 8.2.3)</td>
</tr>
</tbody>
</table>

\(\dagger\) The \texttt{R}-documentation gives the full list of arguments.

Furthermore, Table 3.2.1 describes some relevant arguments of \texttt{vglm()}, while the following output gives the full list:

```r
> args(vglm)

function (formula, family = stop("argument 'family' needs to be assigned"),
  data = list(), weights = NULL, subset = NULL, na.action = na.fail,
  etastart = NULL, mustart = NULL, coefstart = NULL, control = vglm.control(...),
  offset = NULL, method = "vglm.fit", model = FALSE, x.arg = TRUE,
  y.arg = TRUE, contrasts = NULL, constraints = NULL, extra = list(),
  form2 = NULL, qr.arg = TRUE, smart = TRUE, ...) NULL
```

Also developed under the S4–OOP system, \texttt{VGAMextra} extends the scope and functionalities of \texttt{VGAM} in a few directions. At present, \texttt{VGAMextra} supplies a range of additional functions and methods allowing VGLMs/VGAMs to handle other data types, including:
3.3. New family functions for VGLTSMs

Table 3.3.1. Summary of central VGLTSMffs currently available in VGAMextra. Some arguments are shown.

<table>
<thead>
<tr>
<th>VGLTSM</th>
<th>VGLTSMffs derived(^\d)</th>
<th>Default linear Predictor (viz (2.2.4))</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGLM–ARIMAX</td>
<td>ARXff((\text{order} = u))</td>
<td>(\eta_t = \left(\mu_t, \log \sigma^2_{\epsilon_t</td>
</tr>
<tr>
<td>(Section 2.2.1)</td>
<td>MAXff((\text{order} = v))</td>
<td>(\eta_t = \left(\mu_t, \log \sigma^2_{\epsilon_t</td>
</tr>
<tr>
<td></td>
<td>ARMAXff((\text{order} = c(u, v))), and ARIMAXff((\text{order} = c(u, d, v)))</td>
<td>(\eta_t = \left(\mu_t, \log \sigma^2_{\epsilon_t</td>
</tr>
<tr>
<td></td>
<td>ARIMAX.errors.ff((\text{order} = c(u, d, v)))</td>
<td>(\eta_t = \left(\mu_t, \log \sigma^2_{\epsilon_t</td>
</tr>
<tr>
<td>VGLM–ARMAX–GARCH (Section 2.2.2)</td>
<td>ARMAX.GARCH((\text{ARMAorder} = c(u, v))), (\text{GARCHorder} = c(r, s)))</td>
<td>(\eta_t = \left(\mu_t, \log \sigma^2_{\epsilon_t</td>
</tr>
<tr>
<td>VGLM–INGARCH(^\d) (Section 2.2.3)</td>
<td>VGLM.INGARCH((\text{order} = c(u, v)))</td>
<td>(\eta_t = \log \lambda_{t</td>
</tr>
</tbody>
</table>

\(^\d\) "ff" is an abbreviation for “family function”.
\(^\d\) VGLM–INGARCH handles one linear predictor \(\eta_t\), as in (2.2.16), where \(\omega, \vartheta^T, \phi^T, \beta^T_K, \) and \(w^T\) are vector of coefficients that are to be estimated.

- Modelling and estimation of time series with VGLTSMs,
- Quantile modelling of several 1–parameter distributions, as an alternative to quantile regression, and
- Family functions for distributions not included in VGAM, e.g., the multivariate normal, or the generalized Beta of the second kind.

This thesis mainly concentrates on the first, and the following sections provide computational details about the set of functions describing VGLTSMs. Note that various additional aspects may be supressed here due to space limitations, however, the VGAMextra Reference Manual (https://cran.r-project.org/package=VGAMextra) provides complete information on VGLTSMs, VGLTSMffs, and further supplementary material related to its implementation.

3.3 New family functions for VGLTSMs

VGLTSMffs are the sub–class of generic VGLM family functions (VGLMffs; Chapter 18; Yee, 2015) describing VGLTSMs, viz. VGLM–ARIMAX, VGLM–ARMAX–GARCH and VGLM–INGARCH (Sections 2.2.1, 2.2.2, and 2.2.3 respectively). Each VGLTSM– class compresses a number of VGLTSMffs which confer a range of modelling
Table 3.3.2. Some essential slots of VGLTSMffs adopted by typical VGLM–family functions (Table 18.3 of Yee, 2015).

<table>
<thead>
<tr>
<th>Slot</th>
<th>Type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>@blurb</td>
<td>character string</td>
<td>Descriptive. Gives details on the VGLTSMffs.</td>
</tr>
<tr>
<td>@infos</td>
<td>function(...)</td>
<td>Returns a list of data–independent information required by other methods, e.g., <code>summary()</code> or other tests, such as <code>score.stat()</code>.</td>
</tr>
<tr>
<td>@linkinv</td>
<td>function(eta, extra = NULL)</td>
<td>Returns the fitted values, e.g., a matrix with $\mu_T^T$.</td>
</tr>
<tr>
<td>@constraints</td>
<td>expression</td>
<td>Processes the constraint matrices, (cf. (1.3.4)).</td>
</tr>
<tr>
<td>@initialize</td>
<td>expression</td>
<td>Computes the initial values for Fisher scoring. That is, variables <code>etastart</code> or <code>mustart</code>. Performs error checking for variables in <code>formula</code>.</td>
</tr>
<tr>
<td>@loglikelihood</td>
<td>function(mu, y, w, resid = FALSE, eta, extra = NULL, summation = TRUE)</td>
<td>Returns the loglikelihood, computed from the model in turn.</td>
</tr>
<tr>
<td>@vfamily</td>
<td>character string</td>
<td>Assigns the family function identifier.</td>
</tr>
<tr>
<td>@deriv</td>
<td>expression</td>
<td>Returns an $n \times M$ matrix of score vectors (cf. (1.3.13)).</td>
</tr>
<tr>
<td>@weight</td>
<td>expression</td>
<td>Returns the second derivative of the log–likelihood function with respect to the linear predictors (cf. (2.3.4), (1.3.14)). This is the EIM, i.e., Fisher–scoring.</td>
</tr>
<tr>
<td>@last</td>
<td>expression</td>
<td>Information stored on object at final IRLS iteration.</td>
</tr>
</tbody>
</table>

frameworks for time series data that are estimated by MLE using Fisher scoring. Table 3.3.1 shows the central VGLTSMffs currently available in VGAMextra, whereas the following sections describe relevant–internal features making VGLTSMff operational with special emphasis on how these should be manipulated to estimate VGLTSMs and sub–classes.

### 3.3.1 On modelling and estimation of VGLTSMs using VGLTSMffs

As a sub–class of VGLMffs, family functions describing VGLTSMs are objects of class "vglmff" that are similarly assigned to the `family` argument of the modelling function `vglm()`. The typical usage is as follows (`aVGLTSMfamilyfunction` denotes a VGLTSM family function, that are defined in this section):

```r
fit1 <- vglm(y ~ x2 + x3, family = aVGLTSMfamilyfunction, data = tsdata)
```

The essential requirement for a VGLTSM family function is a model that can be estimated by IRLS, that is, a statistical model whose loglikelihood (viz. (2.2.5)), score
Table 3.3.3. Important arguments of VGLTSM family functions derived from VGLM–ARIMAX and VGLM–ARMAX–GARCH.

<table>
<thead>
<tr>
<th>VGLTSMff</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARXff(), MAXff(), ARMAXff()</td>
<td>order</td>
<td>Vector of integers. The ARIMA order ((u, d, v)).</td>
</tr>
<tr>
<td>and ARIMAXff()</td>
<td>zero</td>
<td>Numeric or character string vector. Allows to model certain linear predictors as intercept–only, i.e., (\eta_{t,j} = \eta_j = g_j(\vartheta_j) = \beta_{(j)}).</td>
</tr>
<tr>
<td>type.EIM</td>
<td>What type of EIMs is used in Fisher scoring. Choices are type.EIM = &quot;exact&quot; (viz. (2.3.8)) and type.EIM = &quot;approximate&quot; (viz. (2.3.9)).</td>
<td></td>
</tr>
<tr>
<td>var.arg</td>
<td>Logical. Allows to model either the forecast variance, (\sigma^2_{\varepsilon_t</td>
<td>\varphi_{t-1}}), else the forecast SD, (\sigma_{\varepsilon_t</td>
</tr>
<tr>
<td>noChecks</td>
<td>Logical. Allows to check whether the estimated model is stationary (AR, ARMA), and/or invertible(^\dagger) (MA, ARMA).</td>
<td></td>
</tr>
<tr>
<td>lARcoeff, lMAcoeff</td>
<td>Link functions applied to the standard deviation ((lsd)) or the variance ((lvar)), the AR coefficients, and MA coefficients(^\ddagger) respectively.</td>
<td></td>
</tr>
<tr>
<td>ARMAX.GARCHff()</td>
<td>ARMAorder</td>
<td>Orders ((u, v)) and ((r, s)), of the ARMA / GARCH models for (\mu_{t</td>
</tr>
<tr>
<td>type.GARCH</td>
<td>What variant of GARCH model is imposed on (\sigma^2_{\varepsilon_t</td>
<td>\varphi_{t-1}}). Choices are “ARCH”, “GARCH”, “IGARCH”, “A-ARCH”, “Taylor-Schwert”, “Log-GARCH”, and “M-GARCH”. See Table 2.2.4 for details.</td>
</tr>
<tr>
<td>type.param</td>
<td>What type of parametrization for the variance is to be considered: in terms of past errors, (\varepsilon_{t-r</td>
<td>\varphi_{t-1}}), or past observations ({y_{t-u}; u \geq 1}). See, e.g., Chan et al. (2013), for ARCH models parametrized as per the former.</td>
</tr>
<tr>
<td>G1.transform, G2.transform</td>
<td>Functions (G_1) and (G_2) in (2.2.14).</td>
<td></td>
</tr>
<tr>
<td>Cov.on.Var</td>
<td>Logical. If (\text{TRUE}), then (x_{t(1)}) is included in the model for (\sigma_{\varepsilon_t</td>
<td>\varphi_{t-1}}) (cf (2.2.14)).</td>
</tr>
<tr>
<td>lvar, lsd, lARcoeff, lMAcoeff</td>
<td>Same as with ARMAXff(), above.</td>
<td></td>
</tr>
</tbody>
</table>

\(^\dagger\) By default, lARcoeff, and lMAcoeff equal identitylink. 
\(^\ddagger\) Stationarity and invertibility are verified as per Section 4.4.3.

vector and a closed–form (or approximate) expression for the EIMs can be computed. For VGLTSMs such expressions are given in Section 2.3.1. Also, as objects of the S4–class "vglmff", VGLTSMffs maintain similar coding structure as with VGLM–family functions. Table 3.3.2 enumerates the most relevant slots shared by VGLTSMffs, while Table 18.3 in Yee (2015) gives a comprehensive list of slots comprised by typical VGLM–family functions, and thus by VGLTSM–family functions.
Table 3.3.4. Important arguments of \texttt{VGLM.INGARCHff()}, the VGLTSMff describing the class VGLM–INGARCH.

<table>
<thead>
<tr>
<th>VGLTSMff</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{VGLM.INGARCHff()}</td>
<td>\texttt{order}</td>
<td>Vector of integers. The order–($u, v$) of the INGARCH component.</td>
</tr>
<tr>
<td></td>
<td>\texttt{dist.type}</td>
<td>Distributional model for the response. Current alternatives are &quot;poisson&quot;, &quot;negbinomial&quot;, &quot;logarithmic&quot;, and &quot;yulesimon&quot;.</td>
</tr>
<tr>
<td></td>
<td>\texttt{link}</td>
<td>VGLM–link, $g$, applied to the linear predictor, $\lambda_{t</td>
</tr>
<tr>
<td></td>
<td>\texttt{interventions}</td>
<td>A list with the inputs required to set up intervention analysis. See Section 3.3.5.</td>
</tr>
<tr>
<td></td>
<td>\texttt{f.transform.y}</td>
<td>Should the data be transformed? This is the function $\Omega$ in model (2.2.16).</td>
</tr>
<tr>
<td></td>
<td>\texttt{transform.lambda}</td>
<td>Logical. Allows to apply the VGLM–link, $g$, to past, means, $\Lambda_{t,j_i}$ in model (2.2.16).</td>
</tr>
<tr>
<td></td>
<td>\texttt{lagged.fixed.means}</td>
<td>Logical. Should arbitrary past–means, $U = {i_1, i_2, \ldots, i_u}$, be included in the model $g(\Lambda_{t,j_i})$? (cf. (2.2.2)).</td>
</tr>
<tr>
<td></td>
<td>\texttt{lagged.fixed.obs}</td>
<td>Logical. Should arbitrary past–observations, $V = {j_1, j_2, \ldots, j_v}$, be included in $g(\Lambda_{t,j_i})$?</td>
</tr>
</tbody>
</table>

As a measure of fit at each Fisher scoring iteration, maximizing the log–likelihood is primarily used. The argument in charge of this option is \texttt{criterion}, from \texttt{vglm()}, as follows (default):

```r
fit2 <- vglm(yvector ~ x2 + x3, family = aVGLTSMfamilyfunction, ..., criterion = "loglikelihood", data = tsdata)
```

Alternatively, the regression coefficients may be tested at each IRLS iteration till sufficiently small changes are achieved (Eqn. (8.3) of \textit{Yee}, 2015), as follows:

```r
fit3 <- vglm(yvector ~ x2 + x3, family = aVGLTSMfamilyfunction, ..., criterion = "coefficients", data = tsdata)
```

Here, \texttt{aVGLTSMfamilyfunction} is a VGLTSMff, as in Table 3.3.1, while \texttt{fit2} and \texttt{fit3} will have class "\texttt{vglm}".

Regarding the inputs rendering VGLTSMffs operational, Table 3.3.3 gives a description of the arguments involved with VGLTSMffs from the classes VGLM–ARIMAX and VGLM–ARMAX–GARCH, while Table 3.3.4 does so for VGLM–INGARCH (note that only one VGLTSMff is required here). Except by \texttt{ARIMAXff()} with differencing–order other than zero, all VGLTSMffs return as their fitted values the conditional mean, $E(Y_t|x_t)$, denoted $\mu_T^T$. The fitted values returned by \texttt{ARIMAXff()} for positive differencing–orders correspond indeed to $\hat{y}_t$, rather than the estimated differences, $\hat{\nabla y}_t$. 
3.3. New family functions for VGLTSMs

Table 3.3.5. Schemes adopted to generate initial values \( \eta_t^{(0)} \) in VGLTSMffs.

<table>
<thead>
<tr>
<th>Family function</th>
<th>Special cases</th>
<th>Method of initial values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARIMAXff()</td>
<td>ARMA–GARCH, ARIMA and</td>
<td>A regression of ( y_t ) on lagged ( y_{t-k} ), for large ( k ), to estimate the errors, ( \hat{\varepsilon}_t ).</td>
</tr>
<tr>
<td>ARMAXff() and</td>
<td>ARMA models</td>
<td>Then, a regression of ( y_t ) on lagged values both, ( y_t ) and ( \hat{\varepsilon}_t ).</td>
</tr>
<tr>
<td>ARMAX.GARCHff()</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXff()</td>
<td>The moving average model</td>
<td>A regression of ( y_t ) on lagged ( \hat{\varepsilon}_t ) estimated as with ARMAXff().</td>
</tr>
<tr>
<td>ARXff()</td>
<td>Autoregressive model</td>
<td>The Yule-Walker equations.</td>
</tr>
</tbody>
</table>

Regardless, the fitted values can be retrieved with `fitted(object)`, while the residuals are recovered with `residuals(object, type = "response")`. The VGAMextra Manual provides additional material in this context for VGLTSMffs.

3.3.2 Initial values for VGLTSMffs

Family functions for VGLTSMs are self–starting in a similar fashion as with VGLMffs. This means that starting linear predictors, \( \eta_t^{(0)} \), a vector of length \( T \times M \), are internally computed at the start of the IRLS–Fisher scoring. This is the main purpose of `@initialize`. Importantly, note that the starting linear predictors for VGLTSMffs referred to in this chapter must be distinguished from the initial values required in equation (2.2.8) to compute the usual likelihood in TS models.

Choosing reasonably good starting values is often crucial to the success of the scoring algorithm. Table 3.3.5 gives details on methods implemented in VGLTSM family functions to compute such. Here, the initial \( \eta_t^{(0)} \) are obtained from well–known relationships among the coefficients of the VGLTSM that is subject of analysis. The initial–values schemes adopted by VGLTSMffs provide quite good approximations to the true values, so that, ideally, the algorithm will converge to the optimal MLEs provided the log–likelihood is well–behaved in a neighbourhood of the maximum. Although Fisher–scoring may not converge as fast as the Newton–Raphson algorithm, it is much faster than other popular optimization schemes such as the EM–algorithm. Moreover, recall that the VGLTSM–log–likelihood \( \ell \) is specified in terms of \( \eta_{t,j} = g_j(\theta_{t,j}) \), with \( g_j \) a parameter link function, such that a more quadratic shaped log–likelihood towards the solution may be obtained when choosing appropriate transformations \( g_j \). Indeed, the initial values \( \eta_t^{(0)} \) need not to be too close to the MLEs, since Fisher–scoring involves the EIMs and thus, the parameter space where the working weight matrices are positive–definite is reasonably large.
Alternatively, initial values for the scoring algorithm can also be entered through specific arguments of VGLTSMffs. These are labelled with a pre-fixed letter "i", which stands for initial, and are available for all VGLTSMffs. The exception is \texttt{VGLM.INGARCHff()}, not handling this facility yet, but these will be incorporated over time, e.g., "imu" and "isize". Now, a short description of "i"—labelled arguments available is provided below. Fuller details can be found in the \texttt{VGAMextra} Manual.

(I) "idrift", "imean": Initial values for the intercept. The latter is used for models involving an autoregressive structure in the conditional mean model. The former correspond to TS models with MA component.

(II) "ivar", "isd": Initial value for the conditional variance, \( \sigma^2_{\epsilon_t | \Phi_{t-1}} \). The second choice applies to \( \sigma_{\epsilon_t | \Phi_{t-1}} \), provided \( \text{var.arg} = \text{FALSE} \).

(III) "iARcoeff", "iMAcoeff": Initial values for the AR and MA coefficients.

Furthermore, \texttt{vglm()} offers three general arguments which allow the manual input of initial values, namely \texttt{etastart}, \texttt{coefstart}, and \texttt{mustart}. The initial linear predictors, \( \eta^{(0)}_t \), are then generated as follows. Note that the resulting \( \eta^{(0)}_t \) is whatsoever assigned to the internal variable \texttt{etastart}, arranged as a \( T \times M \) matrix:

(i) Bringing about initial linear predictors \( \eta^{(0)}_t = (\eta^{(0)}_{t,1}, \ldots, \eta^{(0)}_{t,M})^T, 1 \leq t \leq T \), through the argument \texttt{etastart}. Trivally, in this case,

\[
\eta^{(0)}_t = \left( \eta^{(0)}_{1}, \ldots, \eta^{(0)}_{T} \right)^T.
\]

(ii) Entering a set of initial coefficients \( \beta^{(0)} \), that is, the initial iteration in (2.3.1), via \texttt{coefstart}. \( \beta^{(0)} \) should be a \( p_{\text{VLM}} \)-vector (cf. (2.2.7)) and \( H_k \) the associated constraint matrices. Hence,

\[
\eta^{(0)}_t = X_{\text{VLM},T} \cdot \beta^{(0)}
\]

is internally computed, with \( X_{\text{VLM},T} \) as in (2.3.15).

(iii) Choosing the fitted mean, a \( T \)-vector \( \mu^{(0)}_t = E[y_t | x]^{(0)} \), equalling its MLE, if tractable, in which case \( \eta^{(0)}_t = g(\mu^{(0)}_t), 1 \leq t \leq T \), is internally computed in slot \texttt{@linkfun}. The \( \mu^{(0)}_t \) can be entered through the argument \texttt{mustart}.

In practice, (i)–(iii) can be entered as follows:

```r
## For a series of length 'T', and 'M' linear predictors.
my.etastart <- 'A (T x M)--matrix of initial linear predictors'
my.coefstart <- 'A length-(Pvlm x 1) vector of initial coefficients'
##... and only if @linkfun exists, then:
my.mustart <- 'A length-(T x 1) vector of initial fitted means'
```
The preferred choice is (i). Note that (iii) is only applicable if there is a slot @linkfun, but, additionally, \(g(\mu_t^{(0)})\) in (iii) may be undefined for some \(t\), e.g., if \(g = \log\), then \(\log(0)\) for models derived from the VGLM-INGARCH class with response distributed as Poisson. Moreover, the second option heavily depends on the user’s expertise to input good values \(\beta^{(0)}\), yet at the cost of computing \(X_{VLM,T} \cdot \beta^{(0)}\).

Regardless, working with the initial values internally generated by VGLTSMffs upon Table 3.3.5 is the suggested choice, particularly for non–experts, or in presence of little information on the MLEs relative to the model of interest. These have been tested under many different conditions for the parameters involved. Further strategies to choose initial values are provided in Yee (2015, Sections 8.3.1 and 18.3.2), and may be implemented for VGLTSM family functions, e.g., choosing \(\eta_t^{(0)} = g(y_t)\), which often works unless there are singularities at the boundaries of the parameter space.

### 3.3.3 Family functions from the class VGLM–ARIMAX

This section surveys several topics relevant to family functions from the class VGLM–ARIMAX, viz. `ARxff()`, `MAXff()`, `ARMAXff()`, and `ARIMAXff()`, with some examples on their usage. Recall that such VGLTSMffs share a number of arguments, as shown in Table 3.3.3, which however, may affect the analysis differently depending on the TS model considered.

First, the argument `zero`, a family–function–facility to incorporate parameter constraints, particularly useful in presence of covariates. As shown in Table 3.3.3, `zero` may be a character string specifying the linear predictors to be modelled as intercept–only however this section illustrate its usage at an introductory level, as the centre of attention are VGLTSMffs. Fuller details and more advanced examples on `zero` are to be given in Section 4.2.1.

Secondly, VGLTSMffs from the class VGLM–ARIMAX do not intersect. This means that they handle disjoint statistical structures for TS analysis, due to our endeavours to refrain from confusing relationships among VGLTSMffs. For instance, the popular AR(\(p\)) and ARX(\(p\)) models are only handled `ARxff()`. Any attempt to estimate such structures with, e.g., `ARMAXff()`, will halt the scoring algorithm, and a warning will be issued. Likewise, `ARIMAXff()` handles only integrated–ARMA models, while for pure ARMAs, `ARMAXff()` is the choice.
In the following we provide various R code chunks to illustrate operational details on family functions from VGLM–ARIMAX and their arguments. Note that no data is involved. Suppose a series \( \{ y_t \} \) conforms with an order \( (2, 2) \)-ARMA with drift, similar to (1.2.1), say:

\[
\varepsilon_t \sim \mathcal{N}(0, \sigma_{\varepsilon_t}^2), \\
(1 - \vartheta_1 B - \vartheta_2 B^2) y_t = \mu^* + (1 + \phi_1 B + \phi_2 B^2) \varepsilon_t.
\] (3.3.1)

Within the VGLTSM framework, \texttt{ARMAXff()} may estimate (3.3.1) affording a number of modelling alternatives. We show two possibilities:

1. Considering the linear predictor \( \eta_t = (\mu^*, \sigma_{\varepsilon_t}, \vartheta_1, \vartheta_2, \phi_1, \phi_2)^T \), use the following code to simply fit an ARMA(2, 1) to \( \{ y_t \} \):

```r
# my.ts: the series to be analyzed
ts.data <- data.frame(y = my.ts)
fit1.armax <- vglm(y ~ 1, ARMAXff(order = c(2, 2), type.EIM = "exact", 
nomean = FALSE, 
var.arg = FALSE, lvar = "identitylink", 
lARcoeff = "identitylink", # Default 
lMAcoeff = "identitylink", # Default 
# Some examples of initial values 
imean = initial.values.mean, 
isd = initial.values.sd), 
trace = TRUE, data = ts.data)
```

Here, \texttt{var.arg = FALSE} allows \( \sigma_{\varepsilon_t} \) (not \( \sigma_{\varepsilon_t}^2 \)) to be estimated. Note the inclusion of initial values, through \texttt{imean}, and \texttt{isd}.

2. However, instead of \( \sigma_{\varepsilon_t}^2 \), it is very often convenient to model \( \log \sigma_{\varepsilon_t}^2 \), and, say, with no drift–term \( \mu^* \). The linear predictor would have the form:

\[
\eta_t = (\log \sigma_{\varepsilon_t}^2, \vartheta_1, \vartheta_2, \phi_1, \phi_2)^T,
\]

such that (3.3.1) may be estimated with

```r
fit2.armax <- vglm(y ~ 1, ARMAXff(order = c(2, 2), type.EIM = "exact", 
nomean = TRUE, var.arg = TRUE, lvar = "loglink", 
lARcoeff = "identitylink", # Default 
lMAcoeff = "identitylink"), # Default 
trace = TRUE, data = ts.data)
```

The argument \texttt{type.EIM} manages the EIM–type involved in the scoring algorithm. Recall that the choices \texttt{type.EIM = "exact"} and \texttt{type.EIM = "approximate"} are available (Section 2.3.1).
3.3. New family functions for VGLTSMs

Similar to other family functions for VGLTSMs, `ARMAXff()` allows external factors to be incorporated in the analysis, say \( x_{t,(1)} \), where it is interesting to investigate the extent to which \( x_{t,(1)} \) intervenes with the process producing the series \( y_t \). The following are two alternatives using `ARMAXff()`:

1. In the presence of covariates, `ARMAXff()` estimates the ARMA coefficients \( (\vartheta_1, \vartheta_2, \phi_1, \phi_2) \) as well as the conditional mean model \( \mu_t \), as intercept-only, as these are embedded in the conditional variance model. Specially for our example, `ARMAXff()` estimates by default:

\[
\varepsilon_t \sim N(0, \sigma_{\varepsilon_t}^2),
\]

\[
(1 - \vartheta_1 B - \vartheta_2 B^2) y_t = \mu_t + (1 + \phi_1 B + \phi_2 B^2) \varepsilon_t,
\]

\[
\mu_t = \mu^*,
\]

\[
\log \sigma_{\varepsilon_t|\varphi_{t-1}}^2 = \beta_0 + \beta^{T}_{x_{t,(1)}},
\]

which can be estimated by the following:

```r
## Recall that \( x1 \) is reserved for the intercepts.
#data <- transform(data, x2 = x2.a.covariate)
fit3.armax <- vglm(y ~ x2, ARMAXff(order = c(2, 2), type.EIM = "exact",
zero = c("MAcoeff", "ARcoeff", "drift"),
nomean = FALSE, var.arg = TRUE, lvar = "loglink"),
trace = TRUE, data = data)
```

where the linear predictor is \( \eta_t = \left( \mu_t, \log \sigma_{\varepsilon_t|\varphi_{t-1}}^2, \vartheta_1, \vartheta_2, \phi_1, \phi_2 \right)^T \). Note that `ARMAXff()` handles \( \log \sigma_{\varepsilon_t|\varphi_{t-1}}^2 \in \mathbb{R} \) for all \( t \), hence not restriction over \( x_{t,(1)} \), for instance \( x_{t,(1)} > 0 \), is required.

2. Alternatively, we may somewhat investigate the effect of \( x_{t,(1)} \) on the DGP through the conditional mean \( E(Y_t|x_{t,(1)}) = \mu_t \). Thus, compared to (3.3.2), the modelling structure now sees the forecast variance as intercept-only, as follows:

\[
\varepsilon_t \sim N(0, \sigma_{\varepsilon_t}^2),
\]

\[
(1 - \vartheta_1 B - \vartheta_2 B^2) y_t = \mu_t + (1 + \phi_1 B + \phi_2 B^2) \varepsilon_t,
\]

\[
\mu_t = \mu^* + \beta^{T}_{x_{t,(1)}},
\]

\[
\log \sigma_{\varepsilon_t}^2 = \beta_0,
\]

```r
## Alternatively, just:
fit3.armax.bis <- vglm(y ~ x2, ARMAXff(order = c(2, 2),
type.EIM = "exact", nomean = FALSE, var.arg = TRUE, lvar = "loglink"),
trace = TRUE, data = data)
```
involving the same linear predictor as with (3.3.2). Now, the following code is used for estimating (3.3.3):

```r
fit4.armax <-
  vglm(y ~ x2, ARMAXff(order = c(2, 2), type.EIM = "exact",
                     zero = c("MAcoeff", "ARcoeff", "noiseVar"),
                     nomean = FALSE, var.arg = TRUE, lvar = "loglink"),
                     trace = TRUE, ts.data = ts.data)
```

To constrain the effect of \(x_t\) on \(\mu_t\) only, note that the character strings "MAcoeff", "ARcoeff", and "noiseVar" need to be assigned to `zero`, while "mean" (intercept–only) is excluded.

### 3.3.4 Family functions from the class VGLM–ARMAX–GARCH

In a similar manner, this section describes important features of family functions from the class VGLM–ARMAX–GARCH (Section 2.2.2), accompanied by a few examples on its usage.

VGLM–ARMAX–GARCH comprises only one VGLTSMff, `ARMAX.GARCHff()`, which accommodates a range of modelling frameworks conforming with many well–known TS models to investigate conditional heteroskedasticity, such as linear, non–linear, and pure GARCH–like models, some of them presented in Table 2.2.4. At the heart of `ARMAX.GARCHff()` we find modelling alternatives for the conditional variance, \(\sigma^2_{\epsilon_t|\Phi_{t-1}}\), with the ability to incorporate data–transformations in the analysis. The statistical structure handled by `ARMAX.GARCHff()` (viz. (2.2.14)) derives from a sequence of i.i.d. standard Gaussian white noise, denoted \(z_t \overset{i.i.d.}{\sim} \mathcal{N}(0,1)\), as follows:

\[
Y_t|\Phi_{t-1} \sim \mathcal{N}(\mu_t|\Phi_{t-1}, \sigma^2_{\epsilon_t|\Phi_{t-1}}), \tag{3.3.4}
\]

\[
\sigma^2_{\epsilon_t|\Phi_{t-1}} = \omega + \beta'_{2,K} \mathbf{x}_{t,(1)} + \alpha T G_1(\epsilon_{t-r}|\Phi_{t-1}) + \gamma T G_2(\sigma_{\epsilon_{t-r}|\Phi_{t-1}})
\]

\[
\epsilon_{t|\Phi_{t-1}} = z_t \cdot \sigma^2_{\epsilon_t|\Phi_{t-1}},
\]

\[
\mu_{t|\Phi_{t-1}} = \mu_t + \phi^T \mathbf{y}_{t-u} + \phi^T \epsilon_{t-v},
\]

for \(G_1\) and \(G_2\) suitable real–valued functions, where \(\mu_t = \mu^* + \beta'_{1,K} \mathbf{x}_{t,(1)}\) by default when covariates \(\mathbf{x}_{t,(1)}\) are entered. The sequences \(z_t\) and \(\epsilon_{t-k|\Phi_{t-k}}\), \(k \geq 1\) are assumed to be independent.

Along this line, Table 3.3.3 shows part of the arguments that make `ARMAX.GARCHff()` operational. In full, the arguments are:
3.3. New family functions for VGLTSMs

> args(ARMAX.GARCHff)

function (ARMAorder = c(1, 0), GARCHorder = c(1, 0), type.TS = c("ARCH", "GARCH", "IGARCH", "Taylor-Schwert", "A-GARCH", "Log-GARCH", "M-GARCH")[1], type.param = c("residuals", "observed")[1],
  Cov.on.Var = FALSE, noChecks = FALSE, G1.transform = NULL,
  G2.transform = NULL, lvar = NULL, lsd = NULL, ldrift = "identitylink",
  lARcoeff = "identitylink", lMAcoeff = "identitylink", idrift = NULL,
  iARcoeff = NULL, iMAcoeff = NULL)

NULL

Specially, the arguments \( G_1 \) and \( G_2 \) enable the transformations \( G_1 \) and \( G_2 \) in (3.3.4), and the vector–results should not compromise the inherent positivity of the sequences \( \varepsilon_{t-1|\Phi_{t-1}} \) and \( \sigma^2_{t-1|\Phi_{t-1}} \).

Unlike family functions derived from VGLM–ARIMAX, introduced in the preceding section, the argument \( \text{zero} \) is dismissed in \texttt{ARMAX.GARCHff()} and replaced by \texttt{Cov.on.Var}. This is a logical argument which allows the effect of any covariates, say \( \mathbf{x}_{t(1)} \), entered at the \texttt{vglm()} call in the forecast variance model \( \eta_2 = \sigma_{t|\Phi_{t-1}}^2 \). By default, \texttt{Cov.on.Var = FALSE}, that is, the factors \( \mathbf{x}_{t(1)} \) are embedded in the conditional mean model, \( \mu_{t|\Phi_{t-1}} \) only. Set \texttt{Cov.on.Var = TRUE} to incorporate such in the linear predictor handling \( \sigma_{t|\Phi_{t-1}}^2 \).

The following is an example of \texttt{ARMAX.GARCHff()} with a few variants to illustrate its usage. For this, we simulate an ARMAX(1, 1) with ARCH(1) errors, including one covariate jittering the conditional mean model, say:

```r
nn <- 350; warm.up <- 150
arC <- 0.18  # AR coefficient
maC <- 0.25  # MA coefficient
drift <- 1.0  # ARMA intercept
omega <- 0.5  # GARCH intercept
alpha.1 <- 0.20  # ARCH coefficient
x2 <- runif(nn, 1, 3)  # A positive covariate
x2.beta <- 0.28  # x2 coefficient
y <- eps.errors <- numeric(nn)
y[1] <- rnorm(1)
eps.errors[1] <- rnorm(1)
for (jj in 2:nn) {
  eps.errors[jj] <- rnorm(1) * sqrt(omega + alpha.1 * eps.errors[jj - 1]^2)
  y[jj] <- drift + x2.beta[1] * x2[jj] +
    arC * y[jj - 1] + maC * eps.errors[jj - 1] + eps.errors[jj]
```
We firstly fit an intercept–only ARMA(1, 1)–ARCH(1) to the data, i.e., no co-
variates are involved, using `ARMAX.GARCHff()`. This classical ARCH–like statistical
structure is
\[
\varepsilon_t = \sigma_t \phi_{t-1} \cdot z_t, \\
(1 - \theta_1 B) y_t = \mu^* + (1 + \phi_1 B) \varepsilon_t, \\
\sigma_{\varepsilon_t|t-1}^2 = \omega + \alpha_1 \varepsilon_{t-1}^2,
\]
where \( z_t \sim \text{i.i.d.} \sim N(0,1) \). To estimate (3.3.5), showing the estimated coefficients, one can
use the following code (the fitted model is saved as `fit1.garch`):

```r
> fit1.garch <-
    vglm(y ~ 1, ARMAX.GARCHff(armaorder = c(1, 1), garchorder = c(1, 0),
                  type.TS = "ARCH", type.param = "residuals"),
         data = GARCH.ts)
```

Checks on stationarity / invertibility successfully performed.

No roots lying inside the unit circle.

Further details within the 'summary' output.

```r
> coef(fit1.garch, matrix = TRUE) # The estimated coefficients.

    drift.mean1 noiseVar1 ARcoeff11 MAcoeff11
(Intercept) 1.5443 0.566970 0.22036 0.13984
ARCH(1)    0.0000 0.084703 0.00000 0.00000
```

Observe that `noChecks = FALSE` (this is the default), which means that
`ARMAX.GARCHff()` internally verifies whether the AR and MA coefficients give place to
any root inside the unit circle. This argument is fully addressed in Section 4.4.3.

The second model proposed towards (3.3.5) indeed conforms with the DGP used
here, that is, an ARMAX(1, 1), including the factor \( x_{t,2} \), with ARCH(1) errors, viz.:
\[
\varepsilon_t = \sigma_t \phi_{t-1} \cdot z_t, \\
(1 - \theta_1 B) y_t = \mu^* + \beta_1 x_{t,2} + (1 + \phi_1 B) \varepsilon_t, \\
\sigma_{\varepsilon_t|t-1}^2 = \omega + \alpha_1 \varepsilon_{t-1}^2,
\]
which can be estimated within the VGLTSM framework as follows:
New family functions for VGLTSMs

```r
> fit2.garch <-
  vglm(y ~ x2, ARMAX.GARCHff(ARMAorder = c(1, 1), GARCHorder = c(1, 0),
                          Cov.on.Var = FALSE,
                          type.TS = "ARCH", type.param = "residuals"),
                          data = GARCH.ts)
```

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.

```r
> coef(fit2.garch, matrix = TRUE) # The estimated coefficients.
```

<table>
<thead>
<tr>
<th>drift.mean</th>
<th>noiseVar1</th>
<th>ARcoeff1</th>
<th>MAcoeff1</th>
<th>ARCH(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.73899</td>
<td>0.50965</td>
<td>0.22426</td>
<td>0.11861</td>
</tr>
<tr>
<td>x2</td>
<td>0.40391</td>
<td>0.00000</td>
<td>0.00000</td>
<td>0.00000</td>
</tr>
</tbody>
</table>

The central difference between `fit1.garch` and `fit2.garch` is the formula in the `vglm()` call: \( y \sim x_2 \). Here, `Cov.on.Var = FALSE` causes \( x_{t,2} \) to be incorporated in the model for the conditional mean, \( \mu_t \), only.

Lastly, we illustrate the inclusion of transformations in the GARCH component, by fitting a GARCH–variant to our data: the Multiplicative–ARCH(2) (see Table 2.2.4), which has shown to represent a better choice compared to ordinary GARCHs to investigate proportional changes in volatility forecasts, e.g., with seasonal/trend data (see, e.g., Schwert, 1989; Rossi, 1996). The ARMAX(1, 1) model over the conditional mean is preserved. This structure is:

\[
\begin{align*}
\epsilon_t &= \sigma_{t|\phi_{t-1}} \cdot z_t, \\
(1 - \theta_1 B) y_t &= \mu^* + \beta_1 x_{t,2} + (1 + \phi_1 B) \epsilon_t, \\
\log \sigma_{\epsilon_t|\phi_{t-1}}^2 &= \omega + \alpha_1 \log \epsilon_{t-1}^2 + \alpha_2 \log \epsilon_{t-2}^2.
\end{align*}
\]

Model (3.3.7) can be estimated in two ways within the VGLTSM framework: (i) setting `type.TS = "M-GARCH"`, with the specified order, `GARCHorder = c(2, 0)`. This option directly conforms with \( \log \sigma_{\epsilon_t|\phi_{t-1}}^2 = \omega + \alpha_1 \log \epsilon_{t-1}^2 + \alpha_2 \log \epsilon_{t-2}^2 \). Alternatively, (ii) applying the function \( f(x) = \log x \) to the lagged errors \( \epsilon_{t-k}^2, k = 1, 2 \), using `G1.transform`, in a GARCH(2, 0) model. As we are modelling \( \log \sigma_{\epsilon_t|\phi_{t-1}}^2 \), the "loglink"–link is needed here. The code for both choices is given next, where only the estimated coefficients are shown for clarity.

```r
> ## This is option (i)
> fit3.garch <-
  vglm(y ~ x2, ARMAX.GARCHff(ARMAorder = c(1, 1), GARCHorder = c(2, 0),
```
Cov.on.Var = FALSE, noChecks = TRUE,
type.TS = "M-GARCH", type.param = "residuals"),

data = GARCH.ts)

> coef(fit3.garch, matrix = TRUE) # The estimated coefficients.

<table>
<thead>
<tr>
<th>drift.mean1</th>
<th>loglink(noiseVar1)</th>
<th>ARcoeff11</th>
<th>MAcoeff11</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.74691</td>
<td>-0.496238</td>
<td>0.23101</td>
</tr>
<tr>
<td>x2</td>
<td>0.39545</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>M-ARCH(1)</td>
<td>0.00000</td>
<td>0.060576</td>
<td>0.000000</td>
</tr>
<tr>
<td>M-ARCH(2)</td>
<td>0.00000</td>
<td>-0.012593</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

### Alternatively (ii):

> logSq.func <- function(x) 2 * log(x)
> fit4.garch <-

  vglm(y ~ x2, ARMAX.GARCHff(ARMAorder = c(1, 1), GARCHorder = c(2, 0),
    Cov.on.Var = FALSE, lvar = "loglink",
    G1.transform = logSq.func, noChecks = TRUE,
    type.TS = "ARCH", type.param = "residuals"),
    data = GARCH.ts)

> coef(fit4.garch, matrix = TRUE) # The estimated coefficients.

<table>
<thead>
<tr>
<th>drift.mean1</th>
<th>loglink(noiseVar1)</th>
<th>ARcoeff11</th>
<th>MAcoeff11</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.74691</td>
<td>-0.496238</td>
<td>0.23101</td>
</tr>
<tr>
<td>x2</td>
<td>0.39545</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>G1(errors)-(1)</td>
<td>0.00000</td>
<td>0.060576</td>
<td>0.000000</td>
</tr>
<tr>
<td>G1(errors)-(2)</td>
<td>0.00000</td>
<td>-0.012593</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Similarly, suitable transformations for past forecast variances, \( \sigma^2_{\varepsilon_{t-k}|\varphi_{t-k-1}} \), \( k > 1 \), maybe entered through \texttt{G2.transform}. Nevertheless, special care is advised in extreme scenarios. For instance, log–transformations in the presence of zero returns, \( \varepsilon_{t-k} \), due to rounding errors.

### 3.3.5 \texttt{VGLM.INGARCHff()} and intervention analysis

This section gives three examples concerning \texttt{VGLM.INGARCHff()} with a special focus on its arguments and how to conduct intervention analyses (Box and Tiao, 1975; Fokianos and Fried, 2010, 2012). The first two are variants of INGARCHs with no interventions. The third model involves a more complex structure with intervention analysis, so that some artificial data is generated to keep track on how \texttt{VGLM.INGARCHff()} evolves in presence of interventions.

\texttt{VGLM.INGARCHff()} is the (only) VGLTSMff describing the class VGL–INGARCH (viz. (2.2.16)), but it has the ability to accommodate a range of INGARCH–like models
3.3. New family functions for VGLTSMs

for analysis of time series of counts (see Table 2.2.6) via a number of arguments shown in Table 3.3.4. To introduce \texttt{VGLM.INGARCHff()}, we firstly consider two INGARCH–models with no interventions. These are special cases of VGLM–INGARCH:

(i) An Order(2, 2) Log–linear INGARCH (Fokianos and Tjostheim, 2011) exploring some calendar–effects, say, over a 3–time–units period and a 6–time–units period.

The statistical structure, including the linear predictor (Table 2.2.6), is

\[ Y_t|\Phi_{t-1} \sim \mathcal{F}(\lambda_t|\Phi_{t-1}; x_t, \Phi_{t-1}), \]

\[ \log \lambda_t|\Phi_{t-1} = \omega + \vartheta_1 \log(y_{t-3} + 1) + \phi_1 \log \lambda_{t-3} + \vartheta_2 \log(y_{t-6} + 1) + \phi_2 \log \lambda_{t-6}. \]

The distributional options available for the response are: "poisson", "negbinomial", "logarithmic", and "yulesimon", which are managed by the argument dist.type.

Arbitrary lags, e.g., \((i_1, \ldots, i_u)\) and \((j_1, \ldots, j_s)\) as in (2.2.16), allowing \((y_{t-i_1}, \ldots, y_{t-i_u})\) and \((\lambda_{t-j_1}, \ldots, \lambda_{t-j_s})\) in the linear predictor, may be inputted using "lagged.fixed.obs", and "lagged.fixed.means", respectively.

Thus, the following code estimates (3.3.8) with, say \(Y_t|\Phi_{t-1} \sim \text{NB}(\lambda_t|\Phi_{t-1}, p)\):

```r
ingarch.data <- data.frame(y = countsY) # 'countsY' are the data.
myLags <- c(3, 6) # Fixed arbitrary lags.
fit1.ingarch <- vglm(y ~ 1, VGLM.INGARCHff(dist.type = "negbinomial",
                           Order = c(2, 2), link = "loglink",
                           interventions = list(),
                           lagged.fixed.means = my.Lags,
                           lagged.fixed.obs = my.Lags),
                           data = ingarch.data, trace = TRUE)
```

The input for lagged.fixed.means and lagged.fixed.obs allows investigation of side effects of lagged conditional means and lagged observations on the response. The VGLM–link applied to \(\lambda_t|\Phi_{t-1}\) is the natural logarithm.


It may be required to incorporate transformed components of the VGLM–INGARCH linear predictor due to properties such as overdispersion, but permitting other VGLM–link functions for the mean–response, \(\lambda_t|\Phi_{t-1}\). The following is an INGARCH(1, 1) model where the effects of logged–squared past observations plus one external factor, \(x_{t,2}\) over the (actual) expected mean \(\lambda_t|\Phi_{t-1}\) are investigated. This yields:

\[ Y_t|\Phi_{t-1} \sim \mathcal{F}(\lambda_t|\Phi_{t-1}; x_t, \Phi_{t-1}), \]

\[ \lambda_t|\Phi_{t-1} = \omega + \beta_1 x_{t,2} + \vartheta_1 \log(y_{t-1} + 1)^2 + \phi_1 \lambda_{t-1}. \]
Table 3.3.6. Input–types required by the argument `interventions`.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tau</code></td>
<td>Non–negative integer valued vector allocating the <em>times of occurrence</em>, $\tau_i, i = 1, \ldots, s$, (viz. (2.2.17)).</td>
</tr>
<tr>
<td><code>delta</code></td>
<td>Numeric vector of constant (fixed) decay rates, $\delta_i \in [0, 1]$, $i = 1, \ldots, s$.</td>
</tr>
<tr>
<td><code>No.Inter</code></td>
<td>Logical. If <code>FALSE</code> then the second–order interactions (viz. (2.2.18)) are included in the linear predictor, else only single interventions are included, conforming with Fokianos and Fried (2012) and Liboschik et al. (2017).</td>
</tr>
</tbody>
</table>

Data transformations are managed by the argument `f.transform.Y`. The input must be a function–object. Besides, given that only first–lags are considered, `lagged.fixed.means` and `lagged.fixed.obs` are not required, unlike (3.3.8).

Under same assumptions as in (i), (3.3.9) can be estimated by:

```r
my.function <- function(x) 2 * log1p(x)  # Returns log((x + 1)^2)
fitted.ingarch <- vglm(y ~ 1, VGLM.INGARCHff(dist.type = "negbinomial", Order = c(1, 1), link = "identitylink", interventions = list(), lagged.fixed.means = NULL, # Default lagged.fixed.obs = NULL, # Default f.transform.Y = my.function), data = ingarch.data, trace = TRUE)
```

Expressly, `my.function` will be applied to the lagged values $y_{t-1}$, producing $\log(y_{t-1} + 1)^2$. The resultant first–lag is inputted as another term in the regression model for $\lambda_{t|\Phi_{t-1}}$, as required in (3.3.9). Note the use of the VGLM–link "identitylink".

`VGLM.INGARCHff()` also has the ability to conduct intervention analysis, with assistance of the argument `interventions`. This argument is a list with three elements necessarily named as `tau`, `delta`, and `No.Inter`. Table 3.3.6 shows the input–type that must be entered for such. In this respect and to exemplify this VGLTSMff for TSCs, our last INGARCH–like model analyzed is an special case of VGLM–INGARCH class, handled by `VGLM.INGARCHff()` comprising intervention analysis:

(iii) An Order$(1, 0)$ INGARCH with two interventions (NO interactions), 1–lagged effects, and transformed components with covariates (Fokianos and Fried, 2012; Liboschik et al., 2017)
Here, we simulate some data \((n = 100)\) effective for this section only. The underlying model is an Order\((1, 0)\) INGARCH with one–covariate, two interventions (no interactions), including 1–lagged effects, and transformed components. The response is assumed as negative binomial, as with the preceding models.

Say that such interventions occur at time–points \(\tau = (\tau_1, \tau_2)^T\) with effects \(\delta = (\delta_1, \delta_2)^T, \delta_i \in [0, 1]\). Recall that effect–types can be singular, exponential, and permanent (Section 2.2.3). For illustration purposes, we chose \(\delta_1 = 0.5, \delta_2 = 1.0\), occurred at times, say \(\tau_1 = 16\) and \(\tau_2 = 41\).

Such modelling structure has been studied by, e.g., Fokianos and Fried (2012); Liboschik et al. (2017), and has the form:

\[
Y_t | \Phi_{t-1} \sim \mathcal{F}(\lambda_t | \Phi_{t-1}; \mu_t, \Phi_{t-1}), \tag{3.3.10}
\]

\[
\log \lambda_t | \Phi_{t-1} = \omega + \beta_1 x_{t,2} + \theta_1 \log(y_{t-1} + 1)^2 + w_1 \cdot \delta_1^{t-\tau_1} \mathbb{1}(t \geq \tau_1) + w_2 \cdot \delta_2^{t-\tau_2} \mathbb{1}(t \geq \tau_2).
\]

Assuming effect sizes (cf. (2.2.16)), say, \(w_1 = w_2 = 1.0\), the data may be simulated as follows. The plot of the resultant series of counts is shown in Figure 3.3.1.

```r
n <- 200
warm.up <- 100  # Warm-up up values
y <- numeric(n)
x2 <- runif(n)  # A covariate
y[1] <- runif(1, 1, 5)
omega <- 0.5
varthe <- 0.25
del1 <- 0.5      # delta_1
del2 <- 1.0      # delta_2
w1 <- w2 <- 1.0  # Effect sizes (See (2.2.16))
for (ii in 2:n) {
  my.mut <- omega + x2[ii] + varthe * log((y[ii - 1] + 1)^2) + w1 * (del1^(ii - (warm.up + 15))) * ifelse(ii > warm.up + 15, 1, 0) + w2 * (del2^(ii - (warm.up + 40))) * ifelse(ii > warm.up + 40, 1, 0)
  y[ii] <- rnbinom(1, mu = exp(my.mut), size = exp(0))
}
## The 'real' data. Remove warm up values.
y <- y[-c(1:warm.up)]
x2 <- x2[-c(1:warm.up)]
```

Next, we use `VGLM.INGARCHff()` to fit (3.3.10) with the following code. Here, the list of inputs for intervention analysis must be defined in advance, as follows:
Figure 3.3.1. Simulated TSC $\sim$ NB with interventions, based on (3.3.10).

```r
my.list <- list(tau = c(16, 41), delta = c(0.5, 1), No.Inter = TRUE)
ingarch.data <- data.frame(y = y, x2 = x2)
my.function <- function(x) 2 * log1p(x)  # Returns log((x + 1)^2)
fit3.ingarch <- vglm(y ~ x2, VGLM.INGARCHff(dist.type = "negbinomial",
                       Order = c(1, 0), link = "loglink",
                       interventions = my.list,
                       lagged.fixed.means = NULL,
                       lagged.fixed.obs = 1,
                       f.transform.Y = my.function),
                       data = ingarch.data)
```

Specifically, `my.list` exemplifies how to construct the inputted list for the argument `interventions`. The assignment `lagged.fixed.mean = NULL` can be ignored as this is the default. Likewise, `my.function` transforms $y_{t-1}$ as with `fit2.ingarch`. The estimated coefficients are

```r
> # True values: Intercept = 0.5, BX2 = 1, vartheta-y^2_{t-1} = 0.25
> # Effect size-Interv.1 = Effect size-Interv.2 = 1
> t(coef(fit3.ingarch, matrix = TRUE))

  (Intercept) x2 f(Ylag)1 Interv.1 Interv.2
loglink(mu1)   0.93384 1.2194 0.12833 1.0653 1.1922
size1         1.03183 0.0000 0.00000 0.0000 0.0000
```
Table 3.4.1. Complementary VGLTSMff depicted in Section 3.4.

<table>
<thead>
<tr>
<th>VGLTSMff</th>
<th>Addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARff()</td>
<td>Vector autoregression models.</td>
</tr>
<tr>
<td>ARIMAX.errors.ff()</td>
<td>Dynamic regression with ARIMA errors.</td>
</tr>
<tr>
<td>ARMA.studentt.ff()</td>
<td>SARMA–like modelling with Student–t errors.</td>
</tr>
<tr>
<td>ECM.EngleGran()</td>
<td>ECMs for cointegrated time series (Bivariate case).</td>
</tr>
</tbody>
</table>

The effect sizes, \( w_i \), (viz. (2.2.19)) are expected to affect the level of the series at the moment of intervention-occurrence. That is, for instance, by a factor of \( \exp(w_i) \) for log–models using the \texttt{loglink()} link as that above, or by a factor of \( \texttt{logffMeanlink}^{-1}(\omega_i) \) when assuming logarithmic–distributed data and the VGLM–link \texttt{logffMeanlink()} (see Table 1.3.1).

### 3.4 Complementary VGLTSM family functions

In addition to VGLTSMffs from preceding sections and chapters, \texttt{VGAMextra} also comprises a number of implementations addressing other popular modelling frameworks for time series, shown in Table 3.4.1. These results are additional important VGLTSM–modelling–type consequences of the VGLTSM–loglikelihood (2.2.2), also part of this work. Accordingly, in this section we describe the details and modelling features of such VGLTSMffs including directions on its usage, with the exception of \texttt{ECM.EngleGran()} which is an implementation of bivariate ECMs for cointegrated time series, as described in Pfaff (2011, Section 4.2). Due to the extent of ECM–type analysis conferred by this VGLTSMff and connections with other functions in \texttt{VGAMextra}, all the details about this VGLTSMff are presented independently in Chapter 5.

#### 3.4.1 \texttt{VARff()}, a VGLTSMff for VAR–type models

We start by considering \texttt{VARff()}, a VGLTSMff to estimate VAR–type models. \texttt{VARff()} results from the VGLM ability to handle multiple responses, and can be thought of as the multivariate version of \texttt{ARff()}. Its arguments are listed in Table 3.4.2.

Fundamentally, \texttt{VARff()} allows simultaneous exploration of the autocorrelation structure among \( K \) time series, \( \mathbf{y}_t = (y_{t,1}, \ldots, y_{t,K})^T \), assumed to be a finite subset of a \( K \)--dimensional stochastic process realization, say \( \{ \mathbf{Y}_t = (Y_{t,1}, \ldots, Y_{t,k}) ; \mathbf{Y}_t \subseteq \mathbb{R}^K \} \). The statistical structure managed by this family function is the following:
Table 3.4.2. Arguments involved with VARff().

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VAR.order</td>
<td>Numeric; the VAR order. This is $p$ in (3.4.1).</td>
</tr>
<tr>
<td>zero</td>
<td>Character string. Same as in Table 3.3.3.</td>
</tr>
<tr>
<td>lmean, lvar, and lcov</td>
<td>Link functions applied to the conditional means, $\mu_{t</td>
</tr>
</tbody>
</table>

\[
\mathbf{\varepsilon}_t \overset{i.i.d.}{\sim} \text{MNoise}(\mathbf{0}, \mathbf{\Sigma}_t),
\]
\[
Y_{t|\Phi_{t-1}} = \mathbf{\mu}_{t|\Phi_{t-1}} + \mathbf{\varepsilon}_t,
\]
\[
\mathbf{\mu}_{t|\Phi_{t-1}} = \mathbf{\mu}^* + \mathbf{\Gamma} \mathbf{x}_t + \mathbf{\Xi}_1 \mathbf{Y}_{t-1} + \cdots + \mathbf{\Xi}_p \mathbf{Y}_{t-p},
\]
t = 1, \ldots, T. Here, $\mathbf{\varepsilon}_t = (\varepsilon_{t,1}, \ldots, \varepsilon_{t,K})^T$ is a zero mean multivariate random sequence of innovations, while $\mathbf{\Sigma}_t = \mathbb{E}(\mathbf{\varepsilon}_t \cdot \mathbf{\varepsilon}_t^T)$ is the covariance matrix whose $(i,j)^{th}$ entry, denoted $r_{t,(i,j)}$, is given by
\[
r_{t,(i,j)} = \mathbb{E}(\varepsilon_{t,i} \cdot \varepsilon_{t,j}); \quad 1 \leq i, j \leq K.
\]
The $\mathbf{\Xi}_k$ are $K \times K$ matrices of coefficients interconnecting the $K$ series and are to be estimated.

In a similar fashion as with ARff(), covariates $\mathbf{x}_t = (x_{t,1}, \cdots, x_{t,K})^T$ may be included in the analysis, which are embedded in the $j^{th}$ conditional mean model,
\[
\mu_{t,j|\Phi_{t-1}} = \mu_{t,j} = \mathbb{E}(Y_{t,j}|\Phi_{t-1}),
\]
j = 1, \ldots, K, by default. Note that
\[
\mathbf{\mu}_{t|\Phi_{t-1}} = \mathbf{\mu}_t = \mathbb{E}(\mathbf{Y}_t|\Phi_{t-1}) = (\mu_{t,1}, \ldots, \mu_{t,K})^T.
\]
In the latest version, VARff() allows to explore the influence of each $x_{t,j}, 1 \leq j \leq K$ over all equations $y_{t,1}$. Hence, if added, the corresponding coefficients are stacked into $\mathbf{\Gamma}$, a $K \times K$ diagonal matrix, also to be estimated.

The default linear predictor is $(2K + K \cdot (K - 1)/2)$-dimensional and is given by
\[
\eta_t = (\mu_{t,1|\Phi_{t-1}}, \ldots, \mu_{t,K|\Phi_{t-1}}, \log r_{t,(1,1)}, \ldots, \log r_{t,(K,K)}, r_{t,(1,2)}, r_{t,(1,3)} \ldots, r_{t,(K,K+1)})^T.
\]
Here, the "identitylink" is applied to all linear predictors, except for the variances $r_{t,(K,K)}$, where a log–link is applied.
The covariance matrix $\Sigma_t$ is not restricted to be time–invariant, unlike the ordinary VAR model. The covariates $\mathbf{x}_t$ may be directed towards $\Sigma_t$ to explore its effect on the joint variability of the series using the argument zero. By default, zero = c("var", "cov"), resulting in the elements of $\Sigma_t$ being modelled as intercept–only. This means that $\Sigma_t = \Sigma$ is modelled as a Toeplitz matrix (as with the VAR model) with elements as in (3.4.2), however, one may set, say zero = "Mean" to revert this, which allows the inclusion of $\mathbf{x}_t$ in the linear predictors $\eta_t = \log r_{t(i,j)}$. Here, "Mean" is the character string to refer $\mu_t$. Special care is required in this case to ensure that the positive definiteness of $\Sigma_t$ is not compromised in the presence of $\mathbf{x}_t$.

### 3.4.2 ARIMAX.errors.ff() for dynamic regression

Linear schemes of DRMs are also amenable to the VGLTSM framework by means of the VGLTSMff ARIMAX.errors.ff(). DRMs have received special attention since the introduction of the Box–Jenkins methodology (see, e.g. Pankratz, 1991) resulting in a range of modelling choices to investigate dynamic structures on:

(i) time–dependent responses (endogenous), such as the ARX–class of models, or

(ii) on the explanatories (exogenous), like the so–called distributed lag models (see, e.g. Schwartz, 2000), or

(iii) dynamic structures on the error process, where the class of dynamic regression models with ARIMA errors are a prominent example.

VGAMextra::ARIMAX.errors.ff() has the ability to manage DRMs as (ii)–(iii) above. Note that dynamic structures on the response, as in (i), are also manageable by VGLTSMs, specifically through the VGLM–ARIMAX class, and can be studied with VGLTSMffs like ARXff().

However, ARIMAX.errors.ff() is more general. Let $\{y_t; t \in \mathbb{Z}\}$ be a univariate time series from a finite realization of the process $\{Y_t; t \in \mathbb{Z}\}$, and let $\mathbf{x}_{t,(1)}$ be a set of covariates of interest, e.g., interventions or dummy variables, supposed to be linearly associated to the series $y_t$ through the set of coefficients $\beta_{1,K}^T$. At its general, the statistical structure underlying ARIMAX.errors.ff() is two–fold: (i) it accommodates the regression model $\nabla^d Y_t|\Phi_{t-1} \sim \beta_{1,K}^T \nabla^d \mathbf{x}_{t,(1)}$, and (ii) assumes that the errors $\nabla^d u_t = \nabla^d Y_t|\Phi_{t-1} - \beta_{1,K}^T \nabla^d \mathbf{x}_{t,(1)}$ are generated from a non–seasonal ARIMA($p,d,q$) model with random noise $\varepsilon_t$, as follows:

$$
\varepsilon_t \sim \text{Noise}(0, g_1(\sigma^2_{\varepsilon_t|\Phi_{t-1}})),
$$

$$
\vartheta(B) \nabla^d u_t = \phi(B) \varepsilon_t, \quad \text{where}
$$

$$
\nabla^d u_t|\Phi_{t-1} = \nabla^d Y_t|\Phi_{t-1} - \beta_{1,K}^T \nabla^d \mathbf{x}_{t,(1)},
$$

where $g_1(\sigma^2_{\varepsilon_t|\Phi_{t-1}})$ is the conditional variance function of the ARIMA errors.
with \( \vartheta(B), \phi(B) \) as in (2.2.10), and \( g_1 \) a VGLM–link. The resulting model for \( \nabla^d u_t|\Phi_{t-1} \) is commonly referred as an “errors model”.

The default linear predictors are

\[
\eta_t = \left( \mu_{\nabla^d Y_t}, \log \sigma_{\varepsilon_t|\Phi_{t-1}}^2 \right)^T,
\]

where \( \mu_{\nabla^d Y_t} = \mathbb{E}(\nabla^d Y_t|\Phi_{t-1}) \). In general, \texttt{ARIMAX.errors.ff()} allows

\[
g_2(\mu_{\nabla^d Y_t}) = g_2(\mathbb{E}(\nabla^d Y_t|\Phi_{t-1}))
\]

for a VGLM–link \( g_2 \), whose default should clearly be "identitylink" (cf. (3.4.4))

Observe that ARIMA(\( p, d, q \)) errors are primarily assumed. For ARMA(\( u, v \)) errors the model fitted by \texttt{ARIMAX.errors.ff()} accounts for the regression \( Y_t|\Phi_{t-1} \sim \beta_{1,K}^T x_{t,(1)} \), with errors model conforming with

\[
\varepsilon_t \sim \text{Noise}(0, g_1(\sigma_{\varepsilon_t|\Phi_{t-1}}^2)),
\]

\[
\vartheta(B)u_t = \phi(B)\varepsilon_t,
\]

\[
u_t|\Phi_{t-1} = Y_t|\Phi_{t-1} - \beta_{1,K}^T x_{t,(1)}.
\]

Similarly, the default linear predictor is

\[
\eta_t = \left( \mu_{Y_t|\Phi_{t-1}}, \log \sigma_{\varepsilon_t|\Phi_{t-1}}^2 \right)^T,
\]

where \( \mu_{Y_t} = \mathbb{E}(Y_t|\Phi_{t-1}) \) that can be transformed through the VGLM–link \( g_2 \).

The latest version of \texttt{ARIMAX.errors.ff()} estimates (3.4.3)–(3.4.6) for normally distributed innovations \( \varepsilon_t \). A pleasing consequence is that this VGLTSMff is fully operational by MLE–Fisher scoring, and all algorithmic essential like the weight matrices and score vectors are transferred from the class VGLM–ARIMAX, which similarly relies on normal errors. At present only non–seasonal ARIMAs on the errors are handled. This VGLTSMff is currently being upgraded to handle seasonal components, and should be available in \texttt{VGAMextra} in mid 2019.

The central arguments of \texttt{ARIMAX.errors.ff()} are shown in Table 3.4.3. Similar to other VGLTSMffs, explanatories may also be included in the forecast variance model, \( g_2(\sigma_{\varepsilon_t|\Phi_{t-1}}^2) \), through the argument \texttt{zero}. The choices supported by \texttt{zero} in this VGLTSMff are "mean" and "var", standing for \( \mu_{\nabla^d Y_t} \) or \( \mu_{Y_t} \), and \( \sigma_{\varepsilon_t|\Phi_{t-1}}^2 \), respectively. By default, \texttt{zero = "mean"}, hence \( \log \sigma_{\varepsilon_t|\Phi_{t-1}}^2 \) is modelled as intercept–only.

Note particularly that models (3.4.3) and (3.4.5) may allow an intercept represented in the first entry of the vector \( x_{t,(1)} \). Recall that, for VGLTSMs, \( x_{t,1} \equiv 1 \), for all \( t \), and
### Table 3.4.3. Central arguments of ARIMAX.errors.ff().

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>order</td>
<td>Length–3 integer vector. The order($p,d,q$) for the ARIMA model on the errors, $u_t$.</td>
</tr>
<tr>
<td>zero</td>
<td>Character string. Same as in Table 3.3.3, with two choices: &quot;mean&quot; and &quot;var&quot;.</td>
</tr>
<tr>
<td>order.trend</td>
<td>Non–negative integer. If positive, then a polynomial trend of order 'order.trend' is included in the forecast mean model $u_t</td>
</tr>
<tr>
<td>include.int</td>
<td>Logical. Does an intercept should be include in $\mu_{Y_t} (\mu_{\nabla dY_t})$? The default is TRUE.</td>
</tr>
<tr>
<td>diffCovs</td>
<td>Logical. If TRUE (default), all the covariates $x_{t,(1)}$ are differenced up to order $d$ before fitting an ARMA model for the errors in (3.4.3).</td>
</tr>
<tr>
<td>xLag</td>
<td>Non–negative integer. Allows to include order–xLag lags: of either (i) $\nabla^d x_{t,(1)}$ in the model $\nabla^d Y_{t</td>
</tr>
<tr>
<td>lmean, lvar</td>
<td>Link functions applied to $\mu_t$ and $\sigma^2_{\epsilon_{t</td>
</tr>
</tbody>
</table>

Consequently, the intercept in (3.4.3) and (3.4.5) is simply a constant term assumed to be in the models for $\nabla^d Y_{t|\Phi_{t-1}}$ or $Y_{t|\Phi_{t-1}}$. For non–seasonal stationary ARs or ARMAs, for example, such intercept turns out to equal an scaled version of the unconditional mean $E(Y_t) \equiv \mu$, called the drift, as with VGLM–ARIMAX (denoted with $\mu^*$). In the long–run, it can be shown that $\mu^* \to \mu \cdot (1 - \sum_j \vartheta_j)$, for AR coefficients $\vartheta_j$. For moving average schemes, the intercept is consistent with the unconditional mean: $\mu^* = \mu$. This is the meaning when using the argument include.int in ARIMAX.errors.ff() and its interpretation may greatly differ from the meanings conveyed by other software, such as stats::arima() and SAS®.

Linear trend analysis can also be accommodated by this family function through the argument order.trend, which allows to incorporate the polynomial

$$P_t(\gamma) = \sum_{\alpha=0}^{\text{order.trend}} \gamma^\alpha$$

as additional component in the forecast mean model $\mu_{Y_t|\Phi_{t-1}}$ (viz. (3.4.3)). By default order.trend = 0, accounting for an intercept, as conveyed by include.int. Internally, $t^\alpha$, $0 \leq \alpha \leq \text{order.trend}$, are established as time–dependent covariates, hence,
embedded in the vector $x_{t(1)}$ for every $t$–index linear predictor. The coefficients that are to be estimated are added into $\beta_{1,K}^T$. Polynomial trends can be removed by differencing, but non–stationary ARIMAs, say with e.g., $d = 1$, can still have trends if an intercept is preserved, and hence requiring higher order of differencing to fully remove such.

Next, we give some special cases of (3.4.3) and (3.4.5) plus a companion code that may be used to estimate such, using simulated data, with `ARIMAX.errors.ff()`. Alternative software to estimate (3.4.3)–(3.4.5) is available at CRAN, e.g., `stats::arima()` or `forecast::auto.arima()`. Compared to our results, we present the R output obtained from `arima()`, if applicable.

Note that the distributed lag model with ARIMA(1,1,1) errors (viz. (1.1.2)) analysed in Section 1.1 is a sub–class of (3.4.3), and will be conveniently taken as a first example here. Recall that the error series $\{u_t\}$ is modelled as follows:

\[
(1 - \vartheta B) \nabla u_{t|\Phi_{t-1}} = (1 + \phi B) z_t,
\]

(3.4.7)

\[
\nabla u_{t|\Phi_{t-1}} = \nabla y_{t|\Phi_{t-1}} - (\beta_1 \nabla x_{t,2} + \beta_2 \nabla x_{t-1,2}).
\]

To estimate (3.4.7), the following code is proposed, while the output produced by `arima()` is shown in Section 1.1.

```r
## VGLMs handling an LM for E(y_t | x_t), plus an ARIMA(1, 1, 1)
## for the errors
fit.vgam2 <- vglm(y ~ x2, ARIMAX.errors.ff(order = c(1, 1, 1),
                        diffCovs = TRUE, xLag = 1,
                        include.int = FALSE),
                        data = ts.data, trace = TRUE)
coef(fit.vgam2, matrix = TRUE)  ## The estimated coefficients
```

Special attention is needed for the use of the argument `diffCovs`. Table 3.4.3 indicates that `diffCovs = TRUE` is the default and, hence, it must be manually set to `FALSE` in the presence of ARMA–errors to conform with (3.4.3). However, while `diffCovs` may also be set to `FALSE` with ARIMA errors thus resulting in undifferenced covariates, it is highly recommended to keep `diffCovs = TRUE` to attain all the covariates differenced prior the ARIMA fitting. This will greatly help to avoid misleading results such as spurious regression.

From the above R code, for instance, both instructions, `diffCovs = TRUE` and `xLag = 1`, are retained to emphasize the assumption of ARIMA errors and the presence of the first lag, $x_{t-1,2}$, respectively. Setting `diffCovs = FALSE` will result in undifferenced
variables on the right side in the form
\[
\nabla u_{t|\Phi_{t-1}} = \nabla y_{t|\Phi_{t-1}} - (\beta_1 x_{t,2} + \beta_2 x_{t-1,2}),
\]
which will yield unsatisfactory estimates. The omission of the drift, i.e., `include.int = FALSE`, is highly suggested, especially when \( d > 1 \), to refrain from misleading estimates and, very likely, deficient forecasts.

In the next example a time series \((n = 200)\) is generated from a linear regression model with one explanatory, \(x_{t,2} > 0\), and innovations (the error series \(u_t\)) generated from an ARMA(1,1) with, say, zero–intercept. Here, we consider zero–mean Gaussian noise, but its variance is linearly modelled in terms of \(x_{t,2}\) as well. This alternative aims to illustrate the scope of `ARIMAX.errors.ff()`, and especially to show the usage of the argument `zero`. The time–indexed DGP is given by

\[
(1 - \vartheta_1B)u_t = (1 + \phi_1B)\varepsilon_t,
\]

\[
u_t = y_t - (\beta_0 + \beta_1 x_{t,2}),
\]

where \(\varepsilon_t \sim N(0, \sigma^2_{\varepsilon|\phi_{t-1}})\), and \(\sigma^2_{\varepsilon|\phi_{t-1}} = \exp(\omega_0 + \alpha_1 x_{t,2})\).

The proposed code to generate the data upon (3.4.8) is given below. The resulting time series is stored in the variable `TS1`. Alternative software may also be utilized, e.g., `stats::arima()` for the purpose of comparison.

```r
n <- 200

### Coefficients in the residuals, \('e[t]'\), model.
omega0 <- 0.75  # Intercept
alpha.1 <- 0.26  # Coefficient for \(x_{t,2}\)

### Coefficients in the ARMA model for innovations \('u[t]'\).
the.AR <- 0.37  # The AR coefficient
phi.MA <- 0.25  # The MA coefficient

### Coefficients in the regression model, outcome is \('y[t]'\).
beta0 <- 1.5    # Intercept
beta1 <- -0.28  # \(x_{t,2}\) coefficient

# Predictors \('x2'\),
u.errors <- e.errors <- x2 <- TS1 <- runif(n)

for (ii in 2:n) {
    # ...
}
```
Let us assume, firstly, homoskedastic residuals $\varepsilon_t$, that is $\sigma^2_{\varepsilon_t | \Phi_{t-1}} = \exp(\omega_0)$, conserving both models in (3.4.8). Here, we set zero = "var", thus modelling $\sigma^2_{\varepsilon_t | \Phi_{t-1}}$ as intercept–only. The effect of $x_{t,2}$ is restricted to the mean equation only. The following fits such model to $TS1$:

```r
> fit.1.4.6.M1 <- vglm(TS1 ~ x2, ARIMAX.errors.ff(order = c(1, 0, 1), zero = "var", diffCovs = FALSE, lvar = "loglink"), data = ts.data)
> t(coef(fit.1.4.6.M1, matrix = TRUE)) # The estimated coefficients.

(Intercept)       x2        ARcoeff1    MAcoeff1
mean1      1.51881 -0.32653     0.30843    0.33779
loglink(var1) 0.88999 0.00000     0.00000    0.00000
```

The entries diffCovs = FALSE and lvar = FALSE are the default, hence may be dismissed. The estimated coefficients of the ARMA(1, 1) model on the errors are shown in the 3rd–4th columns of the `coef()` call, while the estimated residuals–variance is $\sigma^2_{\varepsilon_t | \Phi_{t-1}} = \exp(0.89) \approx 2.435$. Now, the output produced by `stats::arima()` is the following. The estimates quite conform with our results from ARIMAX.errors.ff, as well as the residuals, whose plot is given in Figure 3.4.1.

```r
> with(ts.data, arima(TS1, order = c(1, 0, 1), xreg = x2))

Call:
  arima(x = TS1, order = c(1, 0, 1), xreg = x2)

Coefficients:
  ar1      ma1     intercept      x2
          0.303  0.334  1.504 -0.328
  s.e.  0.103  0.095  0.257  0.309

sigma^2 estimated as 2.39:  log likelihood = -371.28,   aic = 752.55
```
Secondly, we can fit a model conforming with (3.4.8). For this, set `zero = NULL` to dismiss intercept–only linear predictors. The code for this may be:

```r
> fit.1.4.6.M2 <- vglm(TS1 ~ x2, ARIMAX.errors.ff(order = c(1, 0, 1),
                          zero = NULL),
                         data = ts.data)
> t(coef(fit.1.4.6.M2, matrix = TRUE))  # The estimated coefficients.
   (Intercept)       x2  ARcoeff1  MAcoeff1
mean1          1.52292 -0.33498  0.31244  0.33624
loglink(var1)  0.93916 -0.10280  0.00000  0.00000
```

From the `coef()` output, \( \log \sigma^2_{\epsilon_t | \Psi_{t-1}} = 0.94 - 0.103 \cdot x_{t,2} \) is the estimated model for the forecast variance, our second linear predictor in (3.4.4).

Alternatively, we may explore whether linear trends, say order–1, occur by enabling `order.trend`, as follows:

```r
> fit.1.4.6.M3 <- vglm(TS1 ~ x2, ARIMAX.errors.ff(order = c(1, 0, 1),
                          zero = "var", order.trend = 1),
```

Figure 3.4.1. Residuals from model (3.4.8) estimated with (1) `arima()` and (2) `ARIMAX.errors.ff()`. 

- [Image of residual plots for `arima()` and `ARIMAX.errors.ff()`]
Figure 3.4.2. Residuals from model (3.4.8) estimated with (1) `forecast::Arima()` and (2) `ARIMAX.errors.ff()`.

The VGLTS fitted to TS1 with the above code is

\[(1 - \vartheta B)u_{t|t-1} = (1 + \phi B)z_t, \quad (3.4.9)\]

\[u_{t|t-1} = y_{t|t-1} - (\beta_0 + \beta_1 x_{t,2} + \alpha_1 t).\]

To check for trends in this model, the standard errors from the VGLM object `fit.1.4.6.M3` are now provided. The estimated slopes are expected to be statistically different from zero, if the series is influenced by any of such.

The VGLTS fitted to TS1 with the above code is

\[(1 - \vartheta B)u_{t|t-1} = (1 + \phi B)z_t, \quad (3.4.9)\]

\[u_{t|t-1} = y_{t|t-1} - (\beta_0 + \beta_1 x_{t,2} + \alpha_1 t).\]

To check for trends in this model, the standard errors from the VGLM object `fit.1.4.6.M3` are now provided. The estimated slopes are expected to be statistically different from zero, if the series is influenced by any of such.
The function `forecast::Arima()` is also adept to estimate (3.4.9) by enabling the argument `include.drift`. The output is given next. Here, the estimates “intercept” and “drift” are directly comparable with \((\text{Intercept}):1(\beta_1)\) and \(\text{trend1}(\alpha_1)\) from fit.1.4.6.M3, while the residuals (from both sources) are plotted in Figure 3.4.2.

```r
Arima(ts.data$TS1, order = c(1, 0, 1), include.drift = TRUE, xreg = ts.data$x2)
```

<table>
<thead>
<tr>
<th>Coefficients: a1</th>
<th>ma1</th>
<th>intercept</th>
<th>drift</th>
<th>ts.data$x2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.298</td>
<td>0.336</td>
<td>1.751</td>
<td>-0.003</td>
<td>-0.309</td>
</tr>
<tr>
<td>s.e.</td>
<td>0.103</td>
<td>0.430</td>
<td>0.004</td>
<td>0.311</td>
</tr>
</tbody>
</table>

sigma^2 estimated as 2.45: log likelihood=-371.03
AIC=754.05 AICc=754.49 BIC=773.84

It should be noted that the covariance matrices are estimated in different ways. While the SEs from `ARIMAX.errors.ff()` are computed upon (2.3.14), `forecast::Arima()` mimics `stats::arima()` in this context, and hence estimates this matrix from the Hessian derived from the log–likelihood.

Lastly, assume that an order–(1) distributed lag model suits the forecast mean \(\nabla Y_t|\Phi_{t-1}\) with non–stationary errors, \(u_t\), generated from an ARIMA(1,1,1) model. Also, the sequence of residuals \(\varepsilon_t\) is assumed as Gaussian white noise, hence the second linear predictor is intercept–only.

```r
> fit.1.4.6.M4 <-
  vglm(TS1 ~ x2, ARIMAX.errors.ff(order = c(1, 1, 1), zero = "var",
                                diffCovs = TRUE, xLag = 1,
                                include.int = FALSE),
        data = ts.data, trace = TRUE)
VGLM linear loop 1: loglikelihood = -386.48777
VGLM linear loop 2: loglikelihood = -385.10893
VGLM linear loop 3: loglikelihood = -385.09929
VGLM linear loop 4: loglikelihood = -385.09929
> t(coef(fit.1.4.6.M4, matrix = TRUE)) # The estimated coefficients.
         (Intercept) Diffx2 Diffx2Lag1 ARcoeff1 MAcoeff1
mean1       0.0000 -0.047649  0.28842   0.62745 -0.89103
loglink(var1) 1.0325  0.000000  0.00000   0.00000  0.00000

The intercept for \(\mu_WY_t\) is set to zero and not estimated. The output provided by `arima()` is:
DRMs have been extended to other data types, e.g., Scheike and Martinussen (2006) and its implementation in R, timereg (Scheike and Zhang, 2011), who focused on extensions of the Cox model for survival data by incorporating time-varying effects of explanatory variables, but DRMs are also tractable by inference techniques other than conditional inference, for instance, Bauwens et al. (2000) present a treatment of models for economic time series based on Bayesian inference with an emphasis on dynamic models.

3.4.3 Student–t errors on ARMA modelling using \texttt{ARMA.studentt.ff()} 

Estimation schemes concerning VGLTSM family functions from the classes VGLM–ARIMAX and VGLM–ARMAX–GARCH fundamentally rely on normally distributed errors. This section introduces a variant of VGLM–ARIMAX and its R implementation, the VGLTSMff called \texttt{ARMA.studentt.ff()}, to estimate order($u$, $v$)–ARMA models where the errors–DGP are assumed to be non–central Student–t distributed.

\texttt{ARMA.studentt.ff()} is another outcome from VGLMs, the engine for VGLTSMs, that encompasses TS models outside the limits of the exponential family. The basic framework behind \texttt{ARMA.studentt.ff()} is

\[
\varepsilon_t|\Phi_{t-1} \sim f(\nu_t|\phi_{t-1}, \mu_t|\phi_{t-1}, \sigma_t|\phi_{t-1}), \tag{3.4.10}
\]

\[
\vartheta(B)Y_t|\Phi_{t-1} = \varphi(B)\varepsilon_t|\Phi_{t-1},
\]
Table 3.4.4. Arguments handled by \texttt{ARMA.studentt ff()}. 

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{order}</td>
<td>Two–entries vector, non–negative. The order (u) and (v) of the ARMA model.</td>
</tr>
<tr>
<td>\texttt{zero}</td>
<td>String character. Same as in Table 3.3.3.</td>
</tr>
<tr>
<td>\texttt{cov.Reg}</td>
<td>Logical. If covariates are entered, should these be included in the ARMA model? Default is \texttt{FALSE}</td>
</tr>
<tr>
<td>\texttt{ilocation, lscale, ldf}</td>
<td>Link functions applied to the parameters location, scale, and degrees of freedom.</td>
</tr>
<tr>
<td>\texttt{iscale, idf}</td>
<td>Optional initial values for the parameters. See Section 3.3.2.</td>
</tr>
</tbody>
</table>

where \(f\) denotes the Student–\(t\) density with parameters \(\nu_{t|\Phi_{t-1}}\) (degrees of freedom), \(\mu_{t|\Phi_{t-1}}\) (location), and \(\sigma_{t|\Phi_{t-1}}\) (scale), which is

\[
f(y_t; \nu_{t|\Phi_{t-1}}, \mu_{t|\Phi_{t-1}}, \sigma_{t|\Phi_{t-1}}) = \frac{\Gamma((\nu_t + 1)/2)}{\sqrt{\nu_t\pi}\Gamma(\nu_t/2)} (1 + y_t^2/\nu_t)^{-(\nu_t+1)/2}, \quad \nu_t > 1. \quad (3.4.11)
\]

Due to the nature of the parameters, the linear predictors are by default

\[
\boldsymbol{\eta}_t = (\mu_{t|\Phi_{t-1}}, \log \sigma_{t|\Phi_{t-1}}, \log \log \nu_{t|\Phi_{t-1}})^T. \quad (3.4.12)
\]

The \texttt{loglog()} link, defined as \(\log(\log(\theta))\) for \(\theta > 1\), keeps the degrees of freedom greater than unity.

Table 3.4.4 lists the arguments handled by \texttt{ARMA.studentt ff()}. Specifically, given a time series \(\{y_t\}\), this VGLTSMff estimates the linear predictors \(\boldsymbol{\eta}_t\), plus the ARMA coefficients \(\vartheta_j, j = 1, \ldots, u, \phi_k, k = 1, \ldots, v\) in (3.4.10)–(3.4.11) by Fisher scoring, where, \(\log \log \nu_{t|\Phi_{t-1}}\) is intercept–only by default. This can be overridden through the argument \texttt{zero}, as shown in preceding sections. Note that covariates, if inputted, may be included in the ARMA model (as in dynamic regression models) with \texttt{cov.Reg}.

Estimation schemes for ARMA–like and dynamic models with non–normal errors have received gradual attention over the last decade (Wong et al., 2009; Harvey, 2013; Cheng et al., 2015), perhaps influenced by asymptotic theory anticipating the eventual normality of the residuals, which is achieved in general, for instance with small samples, or with residuals evidencing heavy tails or excessive kurtosis, as in financial modelling or stock market returns (Wong, 2011).
Table 3.4.5. AICs and BICs from models–(i)–(iii).

<table>
<thead>
<tr>
<th>Model</th>
<th>Modelling function</th>
<th>Family function</th>
<th>AIC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i)–(a)</td>
<td>auto.arima()</td>
<td></td>
<td>325.91</td>
<td>345.29</td>
</tr>
<tr>
<td>(i)–(b)</td>
<td>vglm()</td>
<td>ARIMAX.errors.ff()</td>
<td>325.14</td>
<td>344.46</td>
</tr>
<tr>
<td>(ii)</td>
<td>vglm()</td>
<td>ARMAX.ff()</td>
<td>324.36</td>
<td>343.75</td>
</tr>
<tr>
<td>(iii)</td>
<td>vglm()</td>
<td>ARMA.studentt.ff()</td>
<td>297.77</td>
<td>310.69</td>
</tr>
</tbody>
</table>

Under non-normality or heavy-tailed residuals, there exist a number strategies that may be applied to obtain more accurate forecast intervals, e.g., state-space ARIMAs, if the likelihood is intractable, some bias-corrected bootstrapping procedure, or transforming the data at the risk of non-intuitive interpretations. Within R, various options can be employed, such as ARMA.studentt.ff() introduced here, the Bayesian-type package stochvol which handles volatility modelling with conditional innovations sampled from Student-\(t\) (Kastner, 2016), or via a soft solution with forecast::auto.arima() selecting the ARIMA model fitting the best on the data upon the AIC, but still assuming the normality of errors for order selection.

To illustrate the performance of ARMA.studentt.ff(), we will use the data set fpp2::uschange which gives the quarterly time series of personal consumption in the USA, from 1970–2016, along with other socioeconomic-type series. Compared to the following modelling frameworks, we will explore the quarterly changes in US consumption based on ARMA-like models and how the predictor \(Income_t\) influence this representation. But, specifically, we aim to visually assess and compare the aptness of each modelling framework based on determined distributional assumptions.

(i) Dynamic regression with ARIMA errors, using:

(a) forecast::auto.arima(), and

(b) VGAMextra::ARIMAX.errors.ff(),

(iii) ARMAX models with VGAMextra::ARMAX(), and

(iv) ARMA models with Student-\(t\) innovations using AR.studentt.ff().

First, we use auto.arima(), which ranks an ARIMA(1, 0, 2) on the errors as the best AIC-suited. This is the ARMA order to be preserved for (ia)–(iii). The code is as follows.

```r
> ### Model (i)--(a). Using auto.arima()
> fit.uschange1 <- auto.arima(uschange[, "Consumption"],
> xreg = uschange[, "Income"])
> ### Model (i)--(b). Using ARIMAX.errors.ff()
```
3.4. Complementary VGLTS family functions

\[
\hat{\vartheta}(B) Consumption_t = \phi(B) \varepsilon_t,
\]

\[
\begin{align*}
\mu_{(\vartheta_{t-1})} &= \beta_{(1)1}^* + \beta_{(1)2}^* \text{Income}_t, \\
\log \sigma_{(\vartheta_{t-1})} &= \beta_{(2)1}^*, \\
\log \log \nu_{(\vartheta_{t-1})} &= \beta_{(3)1}^*,
\end{align*}
\]

with \(\vartheta(B) = (1 - \vartheta_1 B), \phi(B) = (1 + \phi_1 B + \phi_2 B^2)\). This helps to clarify the differences observed among the AICs/BICs and the trends in the residuals plots.

The QQ–plots exhibit residuals with a moderate long tail extending out to the left which seems to be handled by the profiled Student–\(t\) from \texttt{fit.uschange4} using...
Figure 3.4.3. Residuals computed from (ia) `auto.arima()`, (ib) `ARIMAX.errors.ff()`, (ii) `ARMAX.ff()`, (iii) `ARMA.studentt.ff()`.

ARMA.studentt.ff(). However, determining the extent to which the normal or the Student–t assumptions are violated and what data points chiefly contribute to this is ambiguous.

Alternatively, various formal schemes are available to test lack of fit in time series models, e.g., the Ljung–Box test to verify the residuals resemblance to white noise. Another method for robust inference is the Newey–Wets estimator correcting the covariance matrix on the basis of autocorrelated errors. These are planned to be incorporated over time in VGAMextra.
This section surveys most of the computational details of S3/S4–OOP methods developed for objects generated with VGLTSMffs. These methods represent a crucial step towards the development of model selection and forecasting frameworks for VGLTSMs, that are planned to be implemented at later stages.
In the parlance of VGLMs, VGLTSMffs are functions that describe the statistical models underneath VGLTSMs, such as `VGLM.INGARCHff()` for VGLM–INGARCH, or `ARXff()` for ARX models (class VGLM–ARIMAX). In R, VGLTSMffs are objects of class "vglmff" created under the S4–OOS by calls of the form `new("vglmff", ...),` with a specified signature, a character string VGLTSMff “identifier”, managed by `@vfamily`.

Likewise, when VGLTSMffs are utilised to estimate VGLTSMs using the modelling...
function `VGAM::vglm()`, the outcome is an object of class "vglm", as shown by the following output for `fit.vgam1` (resulting from the fitting of Model (1.1.1)), and for `ARXff()`, which should be an object of class "vgltsmff". Recall that the basic slots representing VGLMffs and VGLTSMffs are depicted in Table 3.3.2.

```r
> class(ARXff())
## ARXff() a VGLTSMff of class "vgltsmff"
[1] "vgltsmff"
attr("package")
[1] "VGAMextra"
> class(fit.vgam1)
## 'fit.vgam1' of class "vglm" created with a 'vgltsmff'
[1] "vglm"
attr("package")
[1] "VGAM"
```

For convenience, objects created from calls of the form `vglm(.., family = VGLTSMff, ..)` will be referred as "vgltsm"–objects, hence distinguishing from ordinary "vglm"–objects. Note that the latter contains the former.

Within the VGLM–framework, many standard generic functions interwined with the S3/S4–OOSs are available and dispatching appropriate methods for "vglm"–objects. See, e.g., Yee (2015, Section 8.4), in particular Tables 8.5, 8.6, 8.7, which gives the list of generic functions that can be applied to "vglm"–objects. A few examples are `fitted()`, `predict()`, and `simulate()`. The latter allows random variates to be generated from VGLMs for each observation at the MLE, that is $y_i$ at $(x_i, \hat{\theta}_{MLE})$. Furthermore, the majority of such are extractor functions, as the following:

(i) `model.matrix(fit, type = "lm")`, and `model.matrix(fit, type = "vlm")` to extract the LM and VLM matrices, $X_{LM}$, $X_{VLM}$, viz. (1.3.11),

(ii) `vcov()`, to extract the estimated covariance matrix, $\hat{\text{Var}}(\hat{\beta}^*)$ (viz. (2.3.14)), and

(iii) `coef()`, `AIC()`, `BIC()`, or `constraints()` which extracts the constraint matrices $H_k$, (see equation (2.2.6)).

It is then implicit that "vgltsm"–objects are also amenable to VGLM–generic functions. However, while many of the dispatched VGLM–methods conform with TS theory, some of them need to be adapted to account for time series quantities. Recall that VGLMs and the VGLM–framework (methods/functions/sub–classes) were assembled on the assumption of non–dependent observations. In this thesis, the VGLM–loglikelihood has been adapted to handle time series, resulting in several VGLM–aspects to be reconciled accordingly. For instance, the expected information matrices for VGLTSMs, where the additive property for independent samples no longer holds.
The development of appropriate S3/S4–OOP methods for VGLTSMs lies at early stages, but finding a few that have been implemented in VGAMextra as part of this work, and are currently operational. The following is the list of such generic functions dispatching suitable methods for "vgltsm"–objects (most of them are VGLM–generic).

1. (S3–OOS) `vglm.control(...)`–like functions: A list of algorithmic constants and parameters for convergence control of VGLTSMffs which is assigned to the @control, e.g., `ARXff.control()`.

2. (S3–OOS) `PIT(...)` for probability calibration using PIT histograms, Section (4.4.1).

3. (S4–OOS) The `summary()` and `show()` generics.

S3/S4–OOS methods (1)–(3) are internally called by generic functions owned by the "vglm"–class, except by `PIT()`, declared as generic in VGAMextra. In relation to (3), only `summary`–methods for "vgltsm"–objects created from VGLTSMffs of the class VGLM–ARIMAX have been implemented at this stage, and are addressed in the following section. The classes VGLM–INGARCH and VGLM–ARMAX–GARCH will be attended correspondingly over time in a similar fashion.

3.5.1 The `summary()` generic for VGLM–ARIMAX

"vgltsm"–objects created from family functions pertaining to VGLM–ARIMAX have the special signature "ARMAvgltsmff". This is the central tag to identify "vgltsm"–objects amenable to the summary method described in this section. Specifically, the `summary()` generic for objects created with VGLTSMffs from the class VGLM–ARIMAX is an S4-OOS method that gives an extended VGLM–summary, which combines inference features from VGLMs and VGLTSMs based on the following three sources. The first source, (a), gives the inference details from VGLMs, while the second and the third components ((b) and (c)) of the summary give the inference details with TS material.

(a) The summary output for "vglm"–objects displayed by the correspondent `show()`.

In essence, this is a 4–column matrix, as follows:

(i) The estimated coefficients, \( \hat{\beta}^*_{(j)k} \), cf. (2.2.7), from linear predictors \( \eta_t \),

(ii) Their standard errors, \( \text{SE} \left( \hat{\beta}^*_{(j)k} \right) \), viz. (2.3.14),

(iii) The Wald statistic, \( X^* = \frac{\left( \hat{\beta}^*_{(j)k} - 0 \right)}{\text{SE} \left( \hat{\beta}^*_{(j)k} \right)} \),

(iv) The 2–sided p–values, testing \( H_0 : \beta^*_{(j)k} = 0 \).
3.5. S3/S4 methods for VGLTSMffs–derived objects

These are accompanied by a set of additional VGLM–associated statistics, such as the residuals degrees of freedom, or relevant tests for MLE–based theory such as the Hauck–Donner effect.

(b) The standard errors of the ARMA estimates computed from the asymptotic distribution of the ARMA process (Brockwell and Davis, 1991; Box et al., 1994).

Under this approach, the ARMA($u$, $v$) process, $\varphi(B)Y_t = \phi(B)\varepsilon_t$, has the following asymptotic distribution:

$$T^{1/2} \left( \hat{\theta} - \theta \right) \implies N \left( 0, \sigma^2_{\varepsilon_t|\varphi_{t-1}\Sigma}^{-1} \right),$$

with $\theta = (\varphi^T, \phi^T)^T$, $\Sigma^{-1}$ a $(u + v) \times (u + v)$ matrix defined by

$$\Sigma = \begin{pmatrix} \Sigma_{uu} & \Sigma_{uv} \\ \Sigma_{eu} & \Sigma_{vv} \end{pmatrix}. \tag{3.5.1}$$

The matrix $\Sigma$ is obtained upon two autoregressive processes, defined as

$$\varphi(B)Y_t = \varepsilon_t, \quad \text{and} \quad \phi(B)X_t = \varepsilon_t, \tag{3.5.2}$$

giving place to $\Sigma_{uu}$, $\Sigma_{ev}$, $\Sigma_{ev}$, and $\Sigma_{eu}$ in (3.5.1). Specifically, $\Sigma_{uu}$, and $\Sigma_{ev}$ are $(u \times u)$ and $(v \times v)$ Toeplitz matrices whose $ij$th elements are $\gamma_Y(i - j) = \text{Cov}(Y_{t-(i-j)}, Y_t)$ and $\gamma_X(i - j) = \text{Cov}(X_{t-(i-j)}, X_t)$ respectively. $\Sigma_{uv}$ and $\Sigma_{vu}$ are $(u \times v)$, $(v \times u)$ Toeplitz matrices with $ij$th entry given by the cross–covariances $\gamma_{XY}(i - j) = \text{Cov}(Y_t, X_{t-(i-j)})$, and $\gamma_{XY}(j - i) = \text{Cov}(Y_t, X_{t-(j-i)})$, respectively, $0 \leq i \leq u$, $0 \leq j \leq v$.

(c) Checks on stationarity and invertibility on the estimated ARMA coefficients.

This is a short summary of the internally performed checks on stationarity and invertibility based on the polynomial roots of the estimated ARMA coefficients, as described in Section 4.4.3. Specifically the roots of $\hat{\varphi}(z) = 0$ and $\hat{\theta}(z) = 0$ from $\hat{\varphi}_{\text{MLE}}$ and $\hat{\theta}_{\text{MLE}}$ are displayed. Any violation to either stationarity or invertibility conditions will result in a warning message.

Statistics from the first source, (a), fully derive from the VGLM–framework (Yee, 2015), including the SEs (a)–(ii) that can be retrieved with vcov(object).

Segment (b) has been implemented for VGLM–ARIMAX, while (c) is operational for VGLM–ARIMAX and VGLM–ARMAX–GARCH (from VGAMextra 0.0-1 onwards) to provide further elements on the analysis of time series. However, for the ARMA coefficients specifically, note that the SEs in (b) are computed upon (2.3.14) for all
\( \hat{\beta}_{(j)k} \), unlike the VGLM–SEs from (a)–(ii) that conform with independent observations. Consequently, the SEs from both sources will very likely differ.

From \texttt{VGAM} 1.0-0 onwards, outputs (a)–(c) were simultaneously displayed by the \texttt{show()} generic function for "\texttt{vglm}"–objects. Here is an example of the extended VGLTSM–summary using \texttt{ARXff()}, where an AR(2) is fitted to some simulated data, and saved as \texttt{fit.AR}. The summary is retrieved with \texttt{summary(fit.AR)}. Note how the three sections above, (a)–(b)–(c), are embedded into the output, as follows: (I) the ordinary VGLM output, which comprises nearly the half–long of the summary, till number of Fisher–scoring iterations is shown, and (II) the additional output with TS substance (this is (b)–(c)), computed by the VGLTSM framework. Additionally, using \texttt{vcov(object)}, the estimated covariance matrix (2.3.14) is also provided.

```r
> n <- 150
> tsdata <- data.frame(x2 = runif(n))  # A single covariate.
> theta1 <- 0.45; theta2 <- 0.31; theta3 <- 0.10  # Coefficients
> drift <- c(1.3, -1.1)  # Two responses.
> sdAR <- c(sqrt(4.5), sqrt(6.0))  # Two responses.
> 
> # Generate two ARs: TS1 and TS2 of orders 2, 3 (Gaussian noise); the drift for 'TS2' depends on x2
> tsdata <- data.frame(tsdata, TS1 = arima.sim(n,
  model = list(ar = c(theta1, theta1^2)), rand.gen = rnorm,
  mean = drift[1], sd = sdAR[1]),
  TS2 = arima.sim(n,
  model = list(ar = c(theta1, theta2, theta3)), rand.gen = rnorm,
  mean = drift[2] * tsdata$x2, sd = sdAR[2]))
>
> # Fitting an AR(2), maximizing the exact log-likelihood. Note that parameter constraints are involved
> # for the true GDP of TS1, but not considered in this fit. "rhobit" is used as link for the AR coeffs.
> fit.AR <- vglm(TS1 ~ 1, ARXff(order = 2, type.EIM = "exact",
  lARcoeff = "rhobit"),
  data = tsdata, trace = TRUE, crit = "loglikelihood")
```

VGLM linear loop 1 : loglikelihood = -323.68799
VGLM linear loop 2 : loglikelihood = -323.68798
VGLM linear loop 3 : loglikelihood = -323.68798

Checks on stationarity / invertibility successfully performed.

Further details within the 'summary' output.

```r
> summary(fit.AR)
```

Call:
\texttt{vglm(formula = TS1 ~ 1, family = ARXff(order = 2, type.EIM = "exact",
1ARcoeff = "rhobit"), data = tsdata, trace = TRUE, crit = "loglikelihood")}

Pearson residuals:

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARdrift1</td>
<td>-3.764</td>
<td>-0.285</td>
<td>0.292</td>
<td>30.79</td>
<td></td>
</tr>
<tr>
<td>loglink(noiseVar1)</td>
<td>-0.707</td>
<td>-0.628</td>
<td>-0.33630</td>
<td>0.318</td>
<td>4.03</td>
</tr>
<tr>
<td>rhobitlink(ARcoeff1)</td>
<td>-11.913</td>
<td>-0.396</td>
<td>-0.01258</td>
<td>0.329</td>
<td>1.81</td>
</tr>
<tr>
<td>rhobitlink(ARcoeff2)</td>
<td>-11.788</td>
<td>-0.405</td>
<td>-0.00703</td>
<td>0.361</td>
<td>9.79</td>
</tr>
</tbody>
</table>
3.5. S3/S4 methods for VGLTSparse-derived objects

Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) |
|----------|------------|---------|----------|
| (Intercept):1 | 1.047 | 0.303 | 3.46 | 0.00054 *** |
| (Intercept):2 | 1.478 | 0.115 | 12.81 | < 2e-16 *** |
| (Intercept):3 | 0.882 | 0.189 | 4.66 | 3.2e-06 *** |
| (Intercept):4 | 0.595 | 0.171 | 3.48 | 0.00050 *** |

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors: 4

Names of linear predictors:
ARdrift1, loglink(noiseVar1), rhobitlink(ARcoeff11), rhobitlink(ARcoeff21)

Log-likelihood: -323.69 on 596 degrees of freedom

Number of iterations: 3

-----

** Standard errors based on the asymptotic distribution of the MLE estimates:

<table>
<thead>
<tr>
<th>ARcoeff1</th>
<th>ARcoeff2</th>
<th>drift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.414</td>
<td>0.289</td>
<td>1.047</td>
</tr>
<tr>
<td>s.e. 0.079</td>
<td>0.079</td>
<td>0.051</td>
</tr>
</tbody>
</table>

Estimated variance (sigma^2) of the errors: 4.384.

Loglikelihood: -323.688
AIC: 655.376, AICc: 655.652, BIC: 667.419

-----

** Summary of checks on stationarity / invertibility:

Polynomial roots of the AR component computed from the estimated coefficients: (Examining stationarity/invertibility)

<table>
<thead>
<tr>
<th>Model1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root1</td>
</tr>
<tr>
<td>Root2</td>
</tr>
</tbody>
</table>

> ### Here is the estimated covariance matrix from the VGLM framework (not adjusted by the time-dependency).
> vcov(fit.AR)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept):1</td>
<td>0.09153807</td>
<td>0.00000000</td>
<td>-0.02146284</td>
<td>-0.01930737</td>
</tr>
<tr>
<td>(Intercept):2</td>
<td>0.00000000</td>
<td>0.01332001</td>
<td>0.00000000</td>
<td>0.00000000</td>
</tr>
<tr>
<td>(Intercept):3</td>
<td>-0.02146284</td>
<td>0.00000000</td>
<td>0.03582806</td>
<td>-0.01907450</td>
</tr>
<tr>
<td>(Intercept):4</td>
<td>-0.01930737</td>
<td>0.00000000</td>
<td>-0.01907450</td>
<td>0.02924669</td>
</tr>
</tbody>
</table>

Segment (b), in particular, confers central advantages over other procedures and software relying on the Hessian matrix to approximate the standard errors and correlations of the coefficient estimators, such as stats::arima(), or forecast::auto.arima(). In cases where the Hessian matrix is not positive definite the SEs–estimating process
is numerically unstable producing deficient SEs. Instead, the theoretical asymptotic co-
variance (3.5.1) gives more stable SEs depending on the process correlation structure
that can be used for confidence limits for the parameters. Nevertheless, this approach
relies on specific conditions such as a large sample size, or the assumption of \( \{ \varepsilon_t \} \)
resembling white noise.

Note that the VGLTSM and VGLM frameworks are also sensitive to non–positive
definiteness up to certain extent, since the estimated covariance matrix (2.3.14) depends
on the Hessian matrix on the MLEs through \( W_{t|t-1}^{-1} \), \( 1 \leq t \leq T \). However, in practice,
both frameworks use the EIMs whose positive–definiteness holds in a much larger
parameter space compared to schemes purely depending on the OIMs.

3.6 Summary

This chapter introduces the set of all VGLTSM family functions that have been im-
plemented in VGAMextra, as a result of the adaptation of the VGLTSM–loglikelihood
to handle TS data. We have also given evidence of the capacity and flexibility of
VGLTSMs to cope with further important models for time series, including multivari-
ate TS (\texttt{VARff()} ) or the first implementation in \( \mathcal{R} \) (to my knowledge) for autoregression
handling non–normal errors, \texttt{ARMA.studentt.ff()}. 

In addition, several technical details on features making VGLTSMs operational
have been described including examples of its usage. Especially, it has been shown and
detailed the extended summary for VGLTSMs, built upon the corresponding S4–OOP
methods implemented for VGLMs. However, the VGLTSM–framework is even broader
and more flexible, and additional material to illustrate this towards modelling TS, as
well as some crucial ideas on the extent of VGLTSMs scope, is the subject of the next
chapter.
Chapter 4

On Supplementary Modelling Features of VGLTSMs

4.1 Introduction

This chapter focusses on two subjects: (i) Supplementary, yet central, modelling features of VGLTSMs by means of VGLMs, and (ii) complementary functions available in VGAMextra to assist the analysis of time series with VGLTSMs. For this, several R–objects internally created when fitting VGLTSMs, but conveying crucial VGLTSM/VGLMs ideas such as constraint matrices, are delineated. In addition, several auxiliary functions rendering VGAMextra operational for IRLS/Fisher scoring are briefly outlined.

4.2 Constrained VGLTSMs

In time series analysis it may be important to explore certain types of linear constraints among the different parameters pertaining VGLTSMs. For example, practitioners may be attracted to investigate the effect of external factors, say \( x_{t,2} > 0 \), on the volatility model, \( g(\sigma_{\epsilon t}^2, \Phi_{t-1}) \), of a series conforming with an ARMA model. Similar to VGLMs, VGLTSMs have been implemented with the ability to handle such parameter constraints via constraint matrices \( H_k \) (viz. (1.3.6) and (2.2.6)) yielding the so–called constrained VGLTSMs. This section gives directions on how to fit this and other constrained VGLTSMs using VGAMextra.

Within the VGLM framework, two ways of fitting constrained VGLMs with respect to \( p \) explanatories, say \( x_t = (x_{t,1}, \ldots, x_{t,p})^T \), are available, as follows:

(i) through family function–specific arguments such as parallel, exchangeable, and zero; and

(ii) by means of the constraints argument from the modelling function VGAM::vglm().
Arguments parallel, exchangeable, and zero represent convenient short-cuts to constrain VGLMs but provide limited flexibility, while the constraints argument allows full flexibility but at the expense of manually setting up each constraint matrix $H_k$. Such arguments are fully shared with VGLTMs allowing similar parameter constraints, however, while parallel and exchangeable may be applicable to TS models, only zero, and constraints are currently operational in VGLTSM-family functions because they have meaning. Specifically, the following subsections give directions on how to set constraints for VGLTSMs using these two arguments, accompanied by some examples of their usage.

4.2.1 Constrained VGLTSMs using the argument zero

Based on Tables 3.3.3 and 3.3.4, within the VGLTSM framework zero may be assigned to any of the following two choices to specify the linear predictors, $\eta_{t,j}$, to be modelled as intercept-only viz. $\eta_{t,j} = \beta^{*\dagger}_{(j)}$: 

(1) A vector of character-strings specifying, fully or partially, the names associated to these linear predictors, or 

(2) alternatively, an integer-valued vector stating their positions in the vector of linear predictors $\eta_t$.

Note that the names and positions referred in (1)–(2) vary among VGLTSMs as described in Chapter 3, and the corresponding R documentation should help in this regard. In particular zero = NULL, along with constraints = NULL, will produce no $\eta_{t,j}$ treated as intercept-only.

Consider the above mentioned example, towards investigating the potential effect of exogenous, say $x_{t,2}, x_{t,3} \in \mathbb{R}$, over the forecast variance of a series $y_t$ conforming, say, with an ARMA(1, 1) model. The code

```r
# Here 'yt' is a time series (response), 'x2', 'x3' covariates.
# This data must be stored in a data frame.
ts.data <- data.frame(y = yt, x2 = xt2, x3 = xt3)
ar.fit <- vglm(y ~ x2 + x3, ARMAXff(order = c(1, 1),
zero = c("drift", "ARcoeff", "MAcoeff"),
lvar = "loglink", varg.arg = TRUE),
data = ts.data, trace = TRUE)
```
would fit the following order–(1, 1, 0, 0) VGLM–ARMAX–GARCH model to the series $y_t$:

$$(1 - \vartheta_1 B)y_t = \mu^* + (1 + \phi_1 B)\varepsilon_t, \quad \varepsilon_t \overset{i.i.d}{\sim} N(0, \sigma_{\varepsilon_t|\Phi_{t-1}}^2). \tag{4.2.1}$$

The call `zero = c("drift", "ARcoeff", "MAcoeff")` constrains the effect of $x_{t,2}$ and $x_{t,3}$ over $\sigma_{\varepsilon_t|\Phi_{t-1}}^2$, such that the linear predictors are specified as follows:

$$
\begin{align*}
\mu^* &= \eta_1 = \beta^*_1 x_{t,1}, \\
\log \sigma_{\varepsilon_t|\Phi_{t-1}}^2 &= \eta_2 = \beta^*_2 x_{t,2} + \beta^*_3 x_{t,3}, \\
\vartheta_1 &= \eta_3 = \beta^*_4, \\
\phi_1 &= \eta_4 = \beta^*_5.
\end{align*}
\tag{4.2.2}
$$

Further, `ARMAXff()` arranges the default vector of linear predictors in the following fashion:

$$
\eta_t = (\mu^*, \log \sigma_{\varepsilon_t|\Phi_{t-1}}^2, \vartheta_1, \phi_1)^T, \tag{4.2.3}
$$

where $\mu^*$, $\vartheta_1$ and $\phi_1$ are placed in the 1st, 3rd, and 4th positions respectively. Thus, the following call will produce equivalent parameter constraints as in (4.2.2), hence fitting the same VGLM–ARMAX(1, 1) as in the previous code chunk:

```
ar.fit.2 <- vglm(y ~ x2 + x3, ARMAXff(order = c(1, 1), zero = c(1, 3, 4), lvar = "loglink", varg.arg = TRUE),
data = ts.data, trace = TRUE)
```

Here are a few comments on both options. First, note that the errors in (4.2.1) resemble Gaussian noise rather than Gaussian white noise due to non–constant structure imposed on the forecast variance model (default in the VGLM–ARIMAX class), unless $x_{t,2}, x_{t,3}$ are constant factors. Second, `ARMAXff()` can handle $\eta_{t,2} = \log \sigma_{\varepsilon_t|\Phi_{t-1}}^2 = \beta^*_2 x_{t,(1)} \in \mathbb{R}$ allowing real–valued explanatories, not necessarily positive, whose effect on the series–volatility is wanted to be explored. Finally, different modelling choices are available for $\eta_{t,2}$, for instance (a) by directly modelling $\sigma_{\varepsilon_t|\Phi_{t-1}}^2$, then `lvar = "identitylink"` is required, or (b) by estimating $\sigma_{\varepsilon_t|\Phi_{t-1}}$, thus allowing $\log \sigma_{\varepsilon_t|\Phi_{t-1}}$. For this set `varg.arg = FALSE`.

Regarding `zero`, it should be noted from (4.2.3) that the linear predictor positions in $\eta_t$ would drastically change upon the order($u, v$) stated in the `vglm()` call, possibly obfuscating its management. For instance, when `order = c(3, 2)` and `nodrift = TRUE`, the AR coefficients would occupy the positions 2, 3, 4, while the MA coefficients would be in the 5th and 6th places. Therefore, managing `zero` as numeric is unsuitable for TS data, such that character–string values appear as better choice as it obliges to enter the correct name(s) of the intercept-only linear predictor(s) with no risk of misspecified positions. Further, while `zero` requires no constraint matrices to be manu-
ally entered, these are internally generated via the function `VGAM::cm.zero.VGAM()` at `ARMAXff()@constraints` where `zero = c("drift", "ARcoeff", "MAcoeff")` is required. In particular, `zero` is ignored in presence of no covariates other than intercepts \( x_{t,1} \equiv 1 \), that is, intercept–only VGLTSMs.

However, `zero` becomes inflexible to cope with analysis where the interest is focused on exploring a set of covariates and their effect on different parameters. In our above example (4.2.1)–(4.2.2), for instance, \( x_{t,2} \) may be influencing \( \sigma^2_{\varepsilon_t|\varphi_{t-1}} \) only, while \( x_{t,3} \) may be related to \( \eta_{t,1} \) only, such that `zero = c("drift", "ARcoeff", "MAcoeff")` or `zero = c(1, 3, 4)` is useless. In such scenarios other options must be considered. Ditto for all VGLTSM family functions from the classes VGLM–ARIMAX and VGLM–ARMAX–GARCH.

Being `zero` a family–function specific argument, the constraint techniques depicted in this subsection are standard for all VGLTSMfns. Complete details associated to every VGLTSMf, such as linear predictors handled by default, are provided in the companion `VGAMextra–Manual`. In a broader sense, VGLM family functions from `VGAM` also accommodate this argument allowing similar parameter constraint techniques. See Yee (2015, Section 3.3.1) for more technical details on this.

4.2.2 Constrained VGLTSMs using `constraints`

The alternative to fit constrained VGLTSMs with respect to \( p \) explanatories \( x_t = (x_{t,1}, \ldots, x_{t,p})^T \) involves the argument `constraints` from the modelling function `VGAM::vglm()`. The input here must be a named `list` containing the constraint matrices \( H_k \) of interest. Recall that matrices \( H_k \)'s specify the desired constraints over the linear predictors at covariate–level, and consequently, \( k \in \{1, \ldots, p\} \), thus giving place to the matrix \( B \) of coefficients (viz. (1.3.6)) to be estimated.

Consider a time series conforming with an order(3) auto–regressive (AR(3) model parametrized with \( \theta_{AR_3} = (\mu^*, \sigma^2_\varepsilon, \vartheta_1, \vartheta_2, \vartheta_3)^T \), viz.

\[
\vartheta(B)y_t = \mu^* + \varepsilon_t, \quad \vartheta(B) = 1 - \vartheta_1 B - \vartheta_2 B^2 - \vartheta_3 B^3, \tag{4.2.4}
\]

with \( \varepsilon_t \sim \text{i.i.d } N(0, \sigma^2_\varepsilon) \), and, for our purposes, also consider the following test:

\[
H_0 : \vartheta_2 = 2\vartheta_1, \quad \text{and} \quad \vartheta_3 = 3\vartheta_1. \tag{4.2.5}
\]

Unlike popular R–software for time series analysis unable to handle (4.2.5), such as `arima()` or `fArma::armaFit()`, the VGLTSM framework offers the option of `constraint` matrices to estimate (4.2.4) and test (4.2.5) simultaneously. Order–(\( p \)) AR models with Gaussian white noise are a special sub–class of VGLM–ARIMAX handled by the family
4.2. Constrained VGLTSMs

When no explanatories are involved in the estimating process, the following are linear predictors associated to (4.2.4) to be modelled as intercept–only:

\[
\begin{align*}
\mu^* &= \eta_1 = \beta_{(1)1}, \\
\log \sigma^2_\varepsilon &= \eta_2 = \beta_{(2)1}, \\
\vartheta_j &= \eta_{j+2} = \beta_{(j+2)1}, \quad j = 1, 2, 3.
\end{align*}
\]  

This AR(3) can be fitted with the following call using VGLTSMs:

```r
## 'yt' is our response.
tsdata <- data.frame(y = yt)
ar.fit2 <- vglm(y ~ 1, ARXff(order = 3, var.arg = TRUE, lvar = "loglink"), data = tsdata)
```

However, to incorporate the restrictions as in (4.2.5) we must account for parameter–specific constraints and find suitable matrices \( H_k \) over the linear predictors, as:

\[
\eta_t = \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \\ \eta_4 \\ \eta_5 \end{pmatrix} = \begin{pmatrix} \mu^* \\ \log \sigma^2_\varepsilon \\ \vartheta_1 \\ \vartheta_2 \\ \vartheta_3 \end{pmatrix} = \begin{pmatrix} \beta_{(1)1}^* \\ \beta_{(2)1}^* \\ \beta_{(3)1}^* \\ \beta_{(4)1}^* \\ \beta_{(5)1}^* \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 3 \end{pmatrix} \cdot \begin{pmatrix} \beta_{(1)1}^* \\ \beta_{(2)1}^* \\ \beta_{(3)1}^* \end{pmatrix},
\]

Hence to test the null hypothesis (4.2.5) for the AR model (4.2.4) using VGLTSMs, we simply enter the matrix \( H_1 \) in the `vglm()` call, as follows:

```r
H1 <- rbind(c(1, 0, 0),
            c(0, 1, 0),
            c(0, 0, 1),
            c(0, 0, 2),
            c(0, 0, 3))
fit.ar.const <- vglm(y ~ 1, ARXff(order = 3, var.arg = TRUE, lvar = "loglink"),
                     constraints = list('(Intercept)' = H1), data = tsdata)
```

Finally, the matrix \( B \) (viz. (1.3.6)) acquires the form

\[
B = \left( H_1 \cdot \beta_{(1)}^* \right) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 2 \\ 0 & 0 & 3 \end{pmatrix} \cdot \begin{pmatrix} \beta_{(1)1}^* \\ \beta_{(2)1}^* \\ \beta_{(3)1}^* \end{pmatrix},
\]  

(4.2.7)
with \( \beta^*_v = (\beta^*_{(1)1}, \beta^*_{(2)1}, \beta^*_{(3)1})^T \), that is, the coefficients to be estimated. Note that the argument zero is ignored in this case because the regression only involves \( x_{t,1} \equiv 1 \), thus intercept-only linear predictors.

A few notes are in order at this stage.

1. VGLM–ARIMAX (Section 2.2.1) has been adapted to confer a flexible framework to handle non–stationarity in the sense of non–constant first and second moments. When \( \eta_{t,2} = \log \sigma^2_{\varepsilon_t(\phi_{t-1})} \) is intercept–only, the random errors resemble white noise, giving place to one special case of VGLM–ARIMAX: the ARMA class.

2. All matrices \( H_k, k = 1, \ldots, p \) must be labelled as per covariates entered in the vglm() call. Particularly, for \( H_1 \), the constraint matrix on the intercepts \( \beta^*_{(k)1} \), a compromise label must be used: \( '(\text{Intercept})' \). For instance, in Section 4.2.2, observe that the constraint matrices are entered in the form

\[
\text{constraints} = \text{list}('(\text{Intercept})' = H1)
\]

where \( H1 \) is order–3 diagonal matrix.

3. Likewise, note that the VGLTSM–fits ar.fit and ar.fit.2 in section 4.2.1 make use of zero to enforce the desired constraints over the parameters. However, both VGLTSMs can be revised using the argument constraints, and simply enter the constraint matrices accordingly. For this, only re–arrange the linear predictors (4.2.2) as

\[
\eta_t = \begin{pmatrix}
\mu^*, \\
\log \sigma^2_{\varepsilon_t(\phi_{t-1})}, \\
\theta_1, \\
\phi_1
\end{pmatrix} = \begin{pmatrix}
\beta^*_{(1)1} \\
\beta^*_{(2)1} + \beta^*_{(2)2} \cdot x_{t,2} \\
\beta^*_{(3)1} \\
\beta^*_{(4)1}
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1
\end{pmatrix} \cdot \begin{pmatrix}
\beta^*_{(1)1} \\
\beta^*_{(2)1} \\
\beta^*_{(3)1} \\
\beta^*_{(4)1}
\end{pmatrix} + \begin{pmatrix}
0 \\
1 \\
0 \\
0
\end{pmatrix} \cdot \beta^*_{(2)2} \cdot x_{t,2}.
\]

Here, \( H_1 \) specifies the constraints on the intercepts \( \beta^*_{(k)1} \), whilst \( H_2 \) does so over \( x_{t,2} \). Thus, one can alternatively explore the effect of \( x_{t,2} \) on the volatility model for (4.2.1)–(4.2.2) with

\[
H1 \leftarrow \text{diag}(4) \quad \# \text{An order--4 identity matrix.}
\]
\[
H2 \leftarrow \text{rbind}(0, 1, 0, 0)
\]

\[
ar.fit.3 \leftarrow \text{vglm}(y \sim x2, \text{ARMAXff}(\text{order} = c(1, 1), \text{zero} = \text{NULL}, \text{lvar} = ''\text{loglink}''),
\]

}\[
\]

\[
\]

\[
\]

\[
\]

\[
\]

\[
\]
4.2. Constrained VGLTSMs

4. However, constructing the matrices $H_k$ as in (4.2.8) may be a bewildering option for beginners. To help, the function `VGAMextra::cm.ARMA()` has been implemented to aid this process. `cm.ARMA()` computes and returns the labelled constraint matrices in several case scenarios. For example, the following computes $H_1$ conforming with (4.2.5) and (4.2.7):

```r
> ( const.h0 <- cm.ARMA(lags.cm = 3) ) # Labelled

$`(Intercept)`

[,1] [,2] [,3]
[1,] 1 0 0
[2,] 0 1 0
[3,] 0 0 1
[4,] 0 0 2
[5,] 0 0 3
```

and then the following readily fits (4.2.4):

```r
fit.ar.const2 <- vglm(y ~ 1, ARXff(order = 3, var.arg = TRUE, lvar = "loglink"),
                          constraints = const.h0,
                          data = data.frame(y = yt))
```

5. Within the VGLM framework, it is possible to allow specific covariates values to affect each linear predictor $\eta_{t,j}$, say $x_{t,kj}$, the ‘$j$th’ value of the $k$th variable for $\eta_{t,j}, x_{t,k},$ and observation $t$. This facility is managed by the `xij` argument, and has shown to have important applications. For instance, in economics, modelling consumer choice through `multinominal()` (Yee, 2010).

At this stage, VGLTSM family functions appear not to need `xij`, however Section 3.4 in Yee (2015) provides further details about this in the VGLM context. Yet, however, the assumption of $\eta_t$–specific covariates values affecting VGLTSM linear predictors seems to be unrealistic.

4.2.3 Constrained VGLTSMs: An order(1)–AR with ARCH(1) errors

Lastly, many popular TS modelling frameworks within the ARCH–GARCH class are special cases of VGLTSMs, where Table 2.2.4 shows a few. A documented example in this context is the re–parametrized AR(1) with ARCH(1)–errors studied by Chan
et al. (2013), given by

$$Y_t = Y_t | \Phi_{t-1} = \alpha Y_{t-1} + \sqrt{\omega + \beta Y_{t-1}^2} \varepsilon_t,$$  \hspace{1cm} (4.2.9)

where $\alpha, \omega > 0$ and $\beta > 0$, are to be estimated, and $\{\varepsilon_t\} \overset{i.i.d.}{\sim} WN(0, 1)$.

The authors proposed a long and technical method to estimate the tail index of $\{Y_t\}$, involving the estimation of (4.2.9) by QMLE. However, (4.2.9) can easily be handled and estimated by VGLTSMs via the VGLM–ARMAX–GARCH class. In fact, since

$$\sigma_{\varepsilon_t | \Phi_{t-1}}^2 = \text{Var}(Y_t | \Phi_{t-1}) = \omega + \beta Y_{t-1}^2,$$

(4.2.9) is fully described by the following VGLTSM:

$$\varepsilon_t \overset{i.i.d.}{\sim} WN(0, 1),$$

$$Y_t | \Phi_{t-1} \sim N(\mu_t | \Phi_{t-1}, \sigma_{\varepsilon_t | \Phi_{t-1}}^2),$$

$$\mu_t | \Phi_{t-1} = \mu^* + \alpha Y_{t-1},$$

$$\sigma_{\varepsilon_t | \Phi_{t-1}}^2 = \text{Var}(Y_t | \Phi_{t-1}) = \omega + \beta Y_{t-1}^2,$$

where VGAMextra offers the following two choices (not exhaustive) to estimate such.

1. First, as an AR(1) with one covariate, i.e., $Y_{t-1}^2$, with (constrained) effect over the variance model.

AR–types are handled by \texttt{ARXff()}, a VGLTSM family function from the VGLM–ARIMAX class. The linear predictors associated to (4.2.10) are

$$\eta_t = \begin{pmatrix} \mu^* \\ \sigma_{\varepsilon_t | \Phi_{t-1}}^2 \\ \alpha \end{pmatrix} = \begin{pmatrix} \beta^*_{(1)1} \\ \beta^*_{(2)1} + \beta^*_{(2)2} Y_{t-1}^2 \\ \beta^*_{(3)1} \end{pmatrix}.$$  \hspace{1cm} \hspace{1cm} (4.2.11)

To constrain the effect of $Y_{t-1}^2$ over $\sigma_{\varepsilon_t | \Phi_{t-1}}^2$, we must find suitable $H_k$’s as

$$\eta_t = \begin{pmatrix} \beta^*_{(1)1} \\ \beta^*_{(2)1} + \beta^*_{(2)2} Y_{t-1}^2 \\ \beta^*_{(3)1} \end{pmatrix} = \begin{pmatrix} \beta^*_{(1)1} \\ \beta^*_{(2)1} \end{pmatrix} + \begin{pmatrix} 0 \\ Y_{t-1}^2 \end{pmatrix} \cdot \beta_{(2)2}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \beta^*_{(1)1} \\ \beta^*_{(2)1} \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \cdot \beta^*_{(2)2} \cdot Y_{t-1}^2,$$  \hspace{1cm} (4.2.11)
giving place to $H_2$, which constrains the effect of $Y_{t-1}^2$ over $\sigma_{\varepsilon_t}\varepsilon_{t-1}$ only. Thus, (4.2.9) is indeed a VGLTS that can be estimated with the following code. Observe that the lagged (squared) values $Y_{t-1}^2$, are computed with the function `VGAMextra::WN.lags()` firstly.

```r
## 'Lag1yt2' are the 1--lagged squared values of yt.
Lag1yt2 <- WN.lags(ts.data$yt, lags = 1)^2
ts.data <- data.frame(yt = ts.data$yt, Lag1yt2 = Lag1yt2)
H1 <- diag(3)
H2 <- rbind(0, 1, 0)
chan.etal <- vglm(yt ~ Lag1yt2, ARXff(order = 1, lvar = "identitylink",
var.arg = TRUE, zero = NULL,
noChecks = TRUE),
constraints = list("(Intercept)" = H1, "Lag1yt2" = H2),
crit = "loglikelihood", data = ts.data)
```

(2) Optionally rather conveniently, (4.2.9) can be seen as an AR(1) model with ARCH(1) errors conditional on $Y_{t-1}^2$.

Linear TS models in the ARMA–GARCH family belong to the VGLM–ARMAX–GARCH class, and are handled by the VGLTSM family function `ARMAX.GARCHff()`. This VGLTSMff accommodates a range of models, based on the arguments shown in Table 3.3.3 and, more detailed, in Section 3.3.4.

To estimate (4.2.9) under this approach, set `type.TS = "ARCH"`, as well as `type.param = "observed"`, leaving the default ARMAX and GARCH orders:

```r
chan.etal2 <-
  vglm(yt ~ 1, ARMAX.GARCHff(ARMAorder = c(1, 0), GARCHorder = c(1, 0),
                     type.TS = "ARCH", type.param = "observed",
                     noChecks = TRUE),
                     data = ts.data)
```

Here, `type.param = "observed"` enables $\sigma_{\varepsilon_t}\varepsilon_{t-1}$ to be conditional on $Y_{t-1}^2$, as in (4.2.9). Otherwise, the choice `type.param = "residuals"` would set $\sigma_{\varepsilon_t}\varepsilon_{t-1} = \omega + \beta\varepsilon_t^2 - 1$.

Unlike option (1), here the constraint matrices $H_1$ and $H_2$ are internally computed by `ARMAX.GARCHff()`. These can be directly retrieved using the generic function `constraints()`, from `VGAM`.
> constraints(chan.etal2)

$`(\text{Intercept})`

\[
\begin{bmatrix}
[1,] & 1 & 0 & 0 \\
[2,] & 0 & 1 & 0 \\
[3,] & 0 & 0 & 1 \\
\end{bmatrix}
\]

$`\text{ARCH(1)}$`

\[
\begin{bmatrix}
[1,] \\
[2,] & 1 \\
[3,] & 0 \\
\end{bmatrix}
\]

Overall, \texttt{ARMAX.GARCHff()} manages a range of ARMA–GARCH–type models (Table 2.2.4) via constraint matrices, which are computed internally in \texttt{first}. It is hoped that \texttt{ARMAX.GARCHff()} will be modified over time to accommodate further special cases of VGLTSMs within the ARMAX–GARCH class, such as the GARCH–in–mean (GARCH–M), which sees the heteroskedasticity term included in the mean equation (Pierre and Eileen, 1998).

### 4.3 The model matrix $X_{LM,T}$

In this section we illustrate the matrix $X_{LM,T}$ for a few of the VGLTSMs, although applicable to all VGLTSMs, using the corresponding VGLTM–family–functions (Table 3.3.1). The matrix $X_{LM,T}$ is internally computed by VGLTM family functions upon the intrinsic features of each VGLTM.

VGLTM are fully characterized by its log–likelihood (2.2.5) by the simple addition of past information into (1.3.8) through $x_{t,p} = x_t$ (viz. (2.2.1)). $X_{LM,T}$ is an $T \times p$ (model) matrix, also called the LM–matrix, constructed from the \texttt{formula} (first argument) of \texttt{vglm()}, as

$$X_{LM,T} = (x_1, x_2, \ldots, x_T)^T.$$  \hfill (4.3.1)

The role of $X_{LM,T}$ is central for VGLTSMs. As with VGLMs, one purpose of $X_{LM,T}$ is to construct $X_{VLM,T}$ as in (2.3.15), thus giving place to inference methods for VGLTSMs described in Section 2.3.2. But, most importantly, $X_{LM,T}$ partly carries that past information allocated in $\Phi_{t-1}$ to be served into the VGLTM–loglikelihood.
4.3 The model matrix $X_{LM,T}$

### 4.3.1 $X_{LM,T}$ of a regression model with ARIMA(1, 1, 1) errors

First, we consider $fit.vgam2$, the regression model with ARIMA(1, 1, 1) errors involving one covariate, $x_{t,2}$ from Section 1.1, defined as:

$$
\nabla y_t|_{\Phi_{t-1}} = \beta_0 + \beta_1 \nabla x_{t,2} + \beta_2 \nabla x_{t-1,2} + \nabla u_t|_{\Phi_{t-1}}, \quad \text{with}
$$

$$
\nabla u_t|_{\Phi_{t-1}} = \vartheta_1 \nabla u_{t-1} + z_t + \phi_1 z_{t-1}.
$$

Note that the first difference of $x_{t,2}$ and the first lag are interwined with the model for $\nabla y_t|_{\Phi_{t-1}}$. The model matrix $X_{LM,T}$ should be retrieved with the extractor function `model.matrix()`, from `VGAM`. This is a methods–function defined for objects of class "vglm", such as `fit.vgam2`. For our purposes, the typical usage is

```r
model.matrix(object, type = "lm", ...)
```

Hence, the following extracts $X_{LM,T}$ from $fit.vgam2$. For clarity, the first six rows are shown:

```r
> head(model.matrix(fit.vgam2, type = "lm"))
     (Intercept) Diffx2 Diffx2Lag1 ARcoeff1 MAcoeff1
 1       1 0.4264609 0.00000000 0.00000000 0.000000e+00
 2       1 0.8029179 0.42646090 -3.96227570 -3.957425e+00
 3       1 1.3414177 0.80291790 1.07102530 -8.715251e-15
 4       1 -2.8261034 1.34141770 0.11866050 2.593371e-02
 5       1 2.7868310 -2.82610340 -1.24999440 -3.261608e+00
 6       1 0.1851423 2.78683100 1.24652040 -3.004332e+00
```

The first column, "(Intercept)", allocates $x_{t,1} \equiv 1$ (cf. (2.2.1)), for $t = 1, \ldots, T$. The columns $ARcoeff1$ and $MAcoeff1$ are the inputs of the ARMA(1, 1) model on $\nabla u_t|_{\Phi_{t-1}}$, that is, $\nabla u_{t-1}$ and $\nabla z_{t-1}$.

### 4.3.2 The $X_{LM,T}$ matrix of an AR(1) model with ARCH(1) errors

Secondly, we consider that AR(1) with ARCH(1) errors studied by Chan et al. (2013) and given in (4.2.9). Continuing from Section 4.2.3, the model matrix $X_{LM,T}$ is similarly retrieved with `model.matrix()`, presented in the following output, in columns "(Intercept)", "ARCH(1)".

```r
> head(cbind(model.matrix(chan.etal2, type = "lm"),
          "Original" = ts.data$Lag1yt2))
     (Intercept) ARCH(1) Original
 1       1 0.00000000 0.00000000
```

4.3 The model matrix $X_{LM,T}$
The column "ARCH(1)" are the squared–lagged values $y_{t-1}^2$ with effect on the conditional variance model only, thus matching (4.2.11), while "(Intercept)" is a series of 1's signifying the intercepts. Our original data is presented side by side in "Original".

### Rows 12 to 18 of the \texttt{X_{\{LM\}}} matrix, to reveal effect--type \texttt{\delta_{1}}

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>x2</th>
<th>f(Ylag)1</th>
<th>Interv.1</th>
<th>Interv.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1</td>
<td>0.26734782</td>
<td>2.772589</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>0.27156987</td>
<td>2.772589</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>0.36834991</td>
<td>0.000000</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.27326519</td>
<td>0.000000</td>
<td>0.00</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0.08750823</td>
<td>0.000000</td>
<td>1.00</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>1</td>
<td>0.92808804</td>
<td>2.772589</td>
<td>0.50</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0.50996485</td>
<td>7.568379</td>
<td>0.25</td>
<td>0</td>
</tr>
</tbody>
</table>
4.3. The model matrix $X_{LM,T}$

> ## Rows 37 to 43 of the $X_{\{LM\}}$ matrix, to reveal effect--type \( \delta_2 \)
> model.matrix(fit3.ingarch, type = "lm")[37:43, ]

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>x2 f(Ylag)1</th>
<th>Interv.1</th>
<th>Interv.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>1</td>
<td>0.7669771</td>
<td>2.197225</td>
<td>4.768372e-07</td>
</tr>
<tr>
<td>38</td>
<td>1</td>
<td>0.3662092</td>
<td>4.795791</td>
<td>2.384186e-07</td>
</tr>
<tr>
<td>39</td>
<td>1</td>
<td>0.9698308</td>
<td>0.000000</td>
<td>1.192093e-07</td>
</tr>
<tr>
<td>40</td>
<td>1</td>
<td>0.6662665</td>
<td>3.218876</td>
<td>5.960464e-08</td>
</tr>
<tr>
<td>41</td>
<td>1</td>
<td>0.5503381</td>
<td>0.000000</td>
<td>2.980232e-08</td>
</tr>
<tr>
<td>42</td>
<td>1</td>
<td>0.6768878</td>
<td>7.377759</td>
<td>1.490116e-08</td>
</tr>
<tr>
<td>43</td>
<td>1</td>
<td>0.1108888</td>
<td>7.110696</td>
<td>7.450581e-09</td>
</tr>
</tbody>
</table>

Note that both interventions are allocated to $\text{Interv.1}$, and $\text{Interv.2}$ as explanatory variables, this is, $(0.5)^{t-16}1(t \geq 16)$ and $(1)^{t-41}1(t \geq 41)$, respectively. From Figure 3.3.1, the second intervention appears to have a gradual significant effect on the series, which shows abrupt changes when $t > 40$, elapsing throughout the reminder.

4.3.4 The VLM matrix $X_{VLM,T}$

Likewise, the function `model.matrix()` allows to extract the so-called VLM matrix $X_{VLM,T}$ (viz. (2.3.15)) from objects of class "vglm", including all VGLTSMs. The only change comes with `type = "vlm"`. For any valid constraint matrices $H_k, k = 1, \ldots, p$, recall that $X_{VLM,T}$ is $(TM) \times p_{VLM}$ with $p_{VLM} = \sum_{k=1}^p ncol(H_k)$ and $M$ the number of linear predictors. The matrix $X_{VLM,T}$ is constructed from $X_{LM,T}$ and is involved with IRLS at many stages, such as the estimation of variance–covariance matrix at Fisher scoring convergence.

The following call shows a snapshot of $X_{VLM,T}$ retrieved from `fit3.ingarch`, which has been estimated with `VGLM.INGARCHff()`.

> head(model.matrix(fit3.ingarch, type = "vlm"))

<table>
<thead>
<tr>
<th></th>
<th>(Intercept):1</th>
<th>(Intercept):2</th>
<th>x2 f(Ylag)1</th>
<th>Interv.1</th>
<th>Interv.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>1</td>
<td>0</td>
<td>0.1487474</td>
<td>0.000000</td>
<td>0 0</td>
</tr>
<tr>
<td>1:2</td>
<td>0</td>
<td>1</td>
<td>0.0000000</td>
<td>0.000000</td>
<td>0 0</td>
</tr>
<tr>
<td>2:1</td>
<td>1</td>
<td>0</td>
<td>0.3488995</td>
<td>2.197225</td>
<td>0 0</td>
</tr>
<tr>
<td>2:2</td>
<td>0</td>
<td>1</td>
<td>0.0000000</td>
<td>0.000000</td>
<td>0 0</td>
</tr>
<tr>
<td>3:1</td>
<td>1</td>
<td>0</td>
<td>0.1997122</td>
<td>4.969813</td>
<td>0 0</td>
</tr>
<tr>
<td>3:2</td>
<td>0</td>
<td>1</td>
<td>0.0000000</td>
<td>0.000000</td>
<td>0 0</td>
</tr>
</tbody>
</table>

For estimation, recall that the response series $y_t$ in `fit3.ingarch` is assumed as negative binomial, which results in two linear predictors: $\log \lambda_{|\Phi_{t-1}}$, and $\log k$, which is the size parameter (intercept–only) of the NB distribution. This is depicted in the output above, where only the even rows, $1:2, 2:2, \text{etc.}$, which are associated to the size
parameter \( k \), result with no–effect from \( x_2 \) and \( f(Y_{\text{lag}}) \), i.e., a zero value, while the odd rows (1:1, 2:1, ...) wear such effects. We finally show the estimated coefficients to emphasize the intercept–only attribute on \( k \), pictured by the VLM matrix.

\[
\begin{array}{cccccc}
\text{(Intercept)} & x_2 & f(Y_{\text{lag}}) & \text{Interv.1} & \text{Interv.2} \\
\text{loglink}(&\mu_1) & 0.9338431 & 1.219374 & 0.1283298 & 1.06532 & 1.192165 \\
\text{size1} & 1.0318316 & 0.000000 & 0.0000000 & 0.00000 & 0.000000 \\
\end{array}
\]

### 4.4 Supporting functions in **VGAMextra**

This section outlines further implementation details of **VGAMextra** to assist on the modelling, analysis and estimation of TS (including counts) with VGLTSMs, such as unit root tests. However, while different implementations of these methodologies may be available at CRAN, this work strives to build **VGAMextra** as a self–contained piece of software, reducing its dependencies on other packages. In addition the **VGAMextra** implementations often have new features not available elsewhere, and most importantly, our future work (Chapter 8) is planned to heavily depend on these functions, such as forecasting methods for VGLTSMs.

#### 4.4.1 PIT plots with **VGAMextra**

The VGLTSM framework comprises one implementation for probabilistic calibration using PIT histograms: the function **VGAMextra**:\( \text{PIT}() \). In essence, \( \text{PIT}() \) computes non–randomized PIT values \( F^*_t \) for a prespecified predictive distribution \( \mathcal{P}(y_t) \) and a TSC, \( \{y_t\} \), based on a sequence \( U = \{u, 0 \leq u \leq 1\} \), as illustrated in Section 1.2.5. Specifically, to examine the adequacy of the distributional assumption towards the data, \( \text{PIT}() \) displays a graph including the histogram of the PITs superimposed on a standard uniform distribution, indicated with a red dashed line. The PITs are expected to follow a uniform distribution if the prespecified distribution is consistent with the DGP. The histogram is constructed upon a series of \( B \) non–overlapping intervals, or bins, with heights given by (1.2.13). At present, \( \text{PIT}() \) is a generic function declared for objects of class "vglm" created with the VLGM–INGARCH family function \textbf{VGLM.INGARCHff()} only. Internally, \( \text{PIT}() \) calls the function \textbf{execute.PIT()}, the actual engine of the PIT methodology adopted in this work.

Table 4.4.1 shows the central arguments involved with \( \text{PIT}() \). The number of bins, \( B \), is handled by the argument \textbf{bins}, where the default is \( B = 12 \), while the the default length of the sequence \( U \) required to compute the sequence \( \mathcal{P}(y_t) \), is \( B + 1 \). Additional graphical arguments from the class "histogram" may be passed on by means of (...),
4.4. Supporting functions in VGAMextra

Table 4.4.1. Central arguments of \texttt{VGAMextra::PIT()}.  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{object}</td>
<td>An object of class &quot;vglm&quot; created by the call \texttt{vglm(...)} using \texttt{VGLM.INGARCH()}.</td>
</tr>
<tr>
<td>\texttt{dist.type}</td>
<td>Character string. Same as in \texttt{VGLM.INGARCHff()} in Table 3.3.4. The cumulative predictive distribution $P_t(\cdot</td>
</tr>
<tr>
<td>\texttt{bins}</td>
<td>(Optional) The number of equally spaced bins, $B$, which the PIT histogram is plotted over. Default is $B = 10$.</td>
</tr>
<tr>
<td>...</td>
<td>Further graphical arguments passed to depict the resulting histogram.</td>
</tr>
</tbody>
</table>

such as other line types (argument \texttt{lty}), or to override default values, e.g., \texttt{xlab}.

Via the argument \texttt{dist.type}, \texttt{PIT()} currently offers four distributional choices for $P(y_t)$ to be assessed by probabilistic calibration: "poisson", "negbinomial", "logarithmic", and "yulesimon". Note that this matches the options handled by \texttt{VGLM.INGARCHff()}. Here, the predictive distribution utilized is attached to the \texttt{object} created after fitting a VGLTSM using \texttt{VGLM.INGARCHff()}. This information is internally extracted by \texttt{PIT()} and then assigned to \texttt{dist.type}. The PIT methodology has been also implemented in the package \texttt{tscount} (Liboschik et al., 2017), via the function \texttt{pit()}. However, this is currently restricted to handle the Poisson and negative binomial distributions.

In the following, we briefly illustrate the usage of \texttt{PIT()} by computing and comparing the PITs derived from an artificial example with simulated data. Consider the occurrence (yes/no) of a (possibly) time–dependent event recorded every hour between 08:00 and 16:00 hours, say, over a 12–month period. For practicity, assume that the expected number of daily events are positively correlated in the very short–term, such that an order(1, 0)–INGARCH structure should be suffice to somewhat model our series, namely:

$$
\log \lambda_t|\Phi_{t-1} = \omega + \beta_1 y_{t-1}.
$$

(4.4.1)

Say that historical records from past years show a baseline mean number of events of $\omega = 2$ events, slowly decaying at $\beta_1 = -0.20$ yearly. For simplicity, no seasonal terms are considered.

Accounting for some degree of overdispersion, this scenario may be reasonably accommodated by the negative binomial distribution, so that we make use of the function \texttt{stats::rnbinom()} to generate random deviates accordingly. The following is the code
for this:

```r
set.seed(20180218)
n <- 565  # Twelve months of daily data plus burn in (200)
y <- numeric(n)  # Our data
size <- 8  # The maximum number of events per day

## Simulating an INGARCH(1, 0) on the mean model.
mu <- numeric(n)
mu[1] <- 4  # Initial mu
y[1] <- rnbinom(1, size = size, mu = mu[1])
omega <- 2.00  # Intercept
beta1 <- -0.20  # AR coefficient
for (jj in 2:n) {
  mu[jj] <- exp(omega + beta1 * y[jj - 1])
y[jj] <- stats::rnbinom(1, size = size, mu = mu[jj])
}

nb.tsdata <- data.frame(y = y[-c(1:200)])  # Removing the burn in data
```

The INGARCH(1,0) model in (4.4.1) is readily handled by `VGLM.INGARCHff()`. The following code chunk then fits a VGLM–INGARCH(1, 0) to the data for two distributional assumptions on the series of counts: (a) negative binomial, and (b) Poisson, giving place to `fit.pit1` and `fit.pit2`, objects of class "vglm".

```r
## With negative binomial.
> fit.pit1 <- vglm(y ~ 1, VGLM.INGARCHff(dist.type = "negbinomial",
                   Order = c(1, 0),
                   link = "loglink"),
                   data = nb.tsdata)
> coef(fit.pit1, matrix = TRUE)  # The estimated coefficients
  loglink(mu1) size1
(Intercept)  1.8560607  6.47894
Ylag1    -0.1772612  0.00000

## Assuming Poisson.
> fit.pit2 <- vglm(y ~ 1, VGLM.INGARCHff(dist.type = "poisson",
                   Order = c(1, 0),
                   link = "loglink"),
                   data = nb.tsdata)
> coef(fit.pit2, matrix = TRUE)  # The esitmated coefficients
  loglink(lambda)
```

4.4. Supporting functions in \textit{VGAMextra}

The corresponding PIT histograms can now be obtained from \texttt{PIT()}, with:

\begin{verbatim}
par(mfrow = c(1,2))
PIT(fit.pit1, bins = 12, main = "", xlab = "Neg. binomial",
    ylim = c(0, 1.5), col = "orange", las = 1)
PIT(fit.pit2, bins = 12, main = "", xlab = "Poisson",
    ylim = c(0, 1.5), col = "orange", las = 1)
\end{verbatim}

Figure 4.4.1 show the resulted PITs, where the red dashed line denotes the standard uniform distribution. Overall, the negative binomial seems to conforms better with the true DGP compared to Poisson, which have the appearance to be inadequately capturing the autocorrelation structure, and hence predicting higher variation than observed (underdispersion).

4.4.2 Unit root tests

URTs are also addressed in \textit{VGAMextra} via the the function \texttt{KPSS.test()}, which is an implementation of the Kwiatkowski–Phillips–Schmidt–Shin (KPSS; Kwiatkowski et al., 1992) test presented in Section 1.2.2. \texttt{KPSS.test()} allows to test the null hypothesis that a univariate series is level or trend stationary, i.e., $H_0$: trend or $H_0$: level ‘stationary’. Its arguments are shown in Table 4.4.2. The truncation lag–parameter is handled by \texttt{"trunc.1"}, with choice \texttt{"short"}, and \texttt{"large"}. The former returns $\iota$ as the smallest integer not less than $3\sqrt{T}/11$, whilst the latter does so with $9\sqrt{T}/11$. 
Table 4.4.2. Arguments involved with \texttt{VGAMextra::KPSS.test()}.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{x}</td>
<td>A univariate time series. An object of class \texttt{ts}.</td>
</tr>
<tr>
<td>\textit{type.H0}</td>
<td>Character. What type of null hypothesis is to be tested. Choices are &quot;level&quot; and &quot;trend&quot;.</td>
</tr>
<tr>
<td>\textit{trunc.l}</td>
<td>Character. To set the truncation lag–parameter, i.e., ( \iota ) in (1.2.7).</td>
</tr>
</tbody>
</table>

The package \texttt{tseries} is also equipped with an implementation to perform the KPSS test: the function \texttt{kpss.test()}. However, while both implementation are quite similar, there is significant difference on how the p–values for each of the tests are computed. \texttt{kpss.test()} uses linear interpolation between the quantiles of the approximate distribution and the vector of corresponding probabilities, at levels 1%, 2.5%, 5%, and 10%, using the function \texttt{approx()}. However, \texttt{approx()} is unable to interpolate beyond the lower and upper limits of the quantile vector entered, hence returning either \texttt{NA} or a bad approximation. On the other hand, \texttt{KPSS.test()}, from \texttt{VGAMextra}, has been implemented to compute the associated p–values via cubic spline interpolations on the vector of quantiles, allowing its extent beyond outside the quantile vector limits. Internally, this is performed by \texttt{pVal.KPSS.test()} (also in \texttt{VGAMextra}), within \texttt{KPSS.test()}. To see this, consider the vector of quantiles from the approximate distribution followed by the test–statistic (1.2.6) when testing trend–stationarity, given in Kwiatkowski et al. (1992): \( q = c(0.119, 0.146, 0.176, 0.216) \), with probabilities \( p = c(0.1, 0.025, 0.05, 0.01) \). For values, say, \( \kappa = c(0.10, 0.176, 0.216, 0.2165, 0.30) \), we now compute the p–values from both approaches:

```r
> # Using linear interpolation, as with tseries::kpss.test()
> pval1 <- approx(x = c(0.119, 0.146, 0.176, 0.216),
                y = c(0.1, 0.025, 0.05, 0.01),
                xout = c(0.10, 0.176, 0.216, 0.228, 0.30))

> # Using cubic splines interpolation, as with VGAMextra::KPSS.test()
> pval2 <- pVal.KPSS.test(x = c(0.119, 0.146, 0.176, 0.216),
                        y = c(0.1, 0.025, 0.05, 0.01),
                        xout = c(0.10, 0.176, 0.216, 0.2165, 0.30))

> # Results:
> t(cbind(tseries =pval1$y, VGAMextra = pval2))

  tseries   NA  0.05  0.01   NA   NA
  VGAMextra 0.2694659 0.05 0.01 0.007651764 1e-16
```
Further implementations concerning unit root tests, e.g., the Dickey–Fuller type and Phillips–like tests, are hoped to be included over time in VGAMextra, as such techniques certainly play a central role for model selection, another short term goal following this work.

4.4.3 Stationarity, stability and invertibility

VGAMextra also comprises an implementation to scan the properties of stationarity, stability and invertibility of ARMA processes: the function `checkTS.VGAMextra()`. Let \( \{Y_t\} \) be a stochastic process obeying the following properties:

(i) \( \{Y_t\} \) satisfies
\[
\vartheta(B)Y_t = \phi(B)\varepsilon_t, \tag{4.4.2}
\]
for some
\[
\vartheta(B) = 1 - \vartheta_1 B - \cdots - \vartheta_u B^u, \quad \phi(B) = 1 + \phi_1 B + \cdots + \phi_v B^v,
\]
with \( \vartheta_u, \phi_v \neq 0 \), and \( \{\varepsilon_t\} \) a white noise process.

(ii) \( \vartheta(B) \) and \( \phi(B) \) have no common factors or, equivalently, the associated characteristics polynomials,
\[
\vartheta(z) = 1 - \vartheta_1 z - \cdots - \vartheta_u z^u, \tag{4.4.3}
\]
\[
\phi(z) = 1 + \phi_1 z + \cdots + \phi_v z^v,
\]
have no common factors.

From above, several nice properties concerning ARMA processes have been established and widely studied, see, e.g., Brockwell and Davis (1991); Box et al. (1994). However, concerning this work, we concentrate on the following:

(a) Upon no common factors of \( \vartheta(B) \) and \( \phi(B) \), the \( p \) roots of \( \vartheta(z) = 0 \) stepping outside the unit circle, i.e.,
\[
|z| = 1 \implies \vartheta(z) \neq 0 \tag{4.4.4}
\]
is a necessary and sufficient condition for (4.4.2) to have a unique stationary solution, given by the following MA(\( \infty \)) representation:
\[
Y_t = \vartheta^{-1}(B)\phi(B)\varepsilon_t = \sum_{j=-\infty}^{\infty} \psi_j \varepsilon_t, \quad \sum_{j=-\infty}^{\infty} |\psi_j| < \infty.
\]
Formally, (i) + (ii) + (a) implies that \( \{Y_t\} \) conforms with an ARMA\((u, v)\) process (or AR\((u)\) accordingly).

(b) The solution achieved in (a) is stable iff the roots of \( \vartheta(z) = 0 \) are outside the unit circle, i.e.,

\[
|z| \leq 1 \implies \vartheta(z) \neq 0. \tag{4.4.5}
\]

Note that satisfying (4.4.5) implies that stationarity conditions (4.4.4) are satisfied, although this is only a necessary condition. Moreover, if \( \{Y_t\} \) is stable, then it has an infinite MA representation, in the form:

\[
Y_t = \sum_{j=0}^{\infty} \psi_j \varepsilon_t, \quad \sum_{j=0}^{\infty} |\psi_j| < \infty. \tag{4.4.6}
\]

(c) \( \{Y_t\} \) is said to be invertible if there is a

\[
\Psi(B) = \psi_0 + \psi_1 B + \psi_2 B^2 + \ldots, \quad \sum_{j=0}^{\infty} |\psi_j| < \infty,
\]

such that \( \varepsilon_t = \Psi(B)Y_t \). This implies that \( \{Y_t\} \) can be expressed in an (equivalent) infinite autoregressive form as:

\[
Y_t = \sum_{i=1}^{\infty} \psi_i Y_{t-i} + \varepsilon_t. \tag{4.4.7}
\]

This property pertain to well–known classes of models for TS, such as the ARs and ARMAs.

Invertibility can be verified through the the polynomial \( \phi(B) \), as follows: \( \{Y_t\} \) is invertible iff the \( q \) roots of \( \phi(z) = 0 \), cf. (4.4.3), lie outside the circle:

\[
|z| \leq 1 \implies \phi(z) \neq 0. \tag{4.4.8}
\]

Specifically, for a given set of coefficients, either \( \vartheta_i, \ i = 1, \ldots, u \) or \( \phi_j, \ j = 1, \ldots, v \), VGAMextra::checkTS.VGAMextra() computes and returns the roots of the polynomials \( \vartheta(z) \) or \( \phi(z) \), as in (4.4.3), hence allowing to verify stationarity, stability and invertibility of the process derived from such coefficients. The arguments handled by checkTS.VGAMextra() are shown in Table 4.4.3.

Currently, checkTS.VGAMextra() collaborates with the VGLTSM framework in two ways, as follows:

(1) With VGLTSM family functions from the classes VGLM–ARIMAX and VGLM–
4.4. Supporting functions in \texttt{VGAMextra}

Table 4.4.3. Arguments involved with \texttt{checkTS.VGAMextra()}.  

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{thetaEst}</td>
<td>The coefficients, $\vartheta_q$ or $\phi_q$, of the polynomials (4.4.3)</td>
</tr>
<tr>
<td>\texttt{tsclass}</td>
<td>Character string. What polynomial should be considered in (4.4.3)? Options are &quot;AR&quot; (then $\vartheta(\zeta)$) and &quot;MA&quot; (then $\phi(\zeta)$)</td>
</tr>
<tr>
<td>\texttt{NofS}</td>
<td>Non–negative integer. The number of processes represented in \texttt{thetaEst}. Default is \texttt{NofS = 1}. Otherwise, the coefficients in \texttt{thetaEst} are accommodated by row (one row per process). The recycling rule may apply.</td>
</tr>
<tr>
<td>\texttt{chOrder}</td>
<td>Positive integer. The order $p$ or $q$ of the corresponding polynomial, as stated by \texttt{tsclass} (cf. (4.4.3)).</td>
</tr>
<tr>
<td>\texttt{retmod}</td>
<td>Logical. The default value of \texttt{TRUE} causes the module of all roots returned, else the raw roots are returned (may be complex numbers).</td>
</tr>
<tr>
<td>\texttt{pRoots}</td>
<td>Logical. The default value of \texttt{TRUE} means that the resulting roots (or moduled roots) are prompted.</td>
</tr>
</tbody>
</table>

ARMAX–GARCH, to verify whether the estimated coefficients conform with stationarity and invertibility condition.

That is, upon the last IRLS–Fisher–scoring iteration, the roots of $\vartheta(\zeta) = 0$ and $\phi(\zeta) = 0$ are internally computed from the estimated coefficients (AR, MA, or ARMA) using \texttt{checkTS.VGAMextra()}, and simultaneously contrasted with (4.4.4), (4.4.8). A message/warning is accordingly issued in the results.

\texttt{VGLTSMffs} endowed with this capability are those given in Table 3.3.1, except by \texttt{ARIMAX.errors.ff()} and \texttt{VGLM.INGARCH()}. Recall that this feature is managed by the argument \texttt{noChecks}.

(2) \texttt{checkTS.VGAMextra} is also involved with summary methods concerning the class \texttt{VGLM–ARIMAX}, as delineated in Section 3.5.

To illustrate the performance of \texttt{checkTS.VGAMextra()}, consider the ARMAX(1, 2) model from Section 3.4.3, saved in \texttt{fit.uschange3}, and fitted with the following code chunk.

```r
fit.uschange3 <- vglm(Consumption ~ Income,
                       ARMAXff(order = c(1, 2), noChecks = TRUE,
                                zero = c("Var", "ARcoeff", "MAcoeff")),
                       data = uschange)
```
Note the `noChecks = TRUE`, hence disabling the corresponding checks, with no output in this line. In the following call we now set `noChecks = FALSE` to show the standard trace/output after an internal check of stationarity and invertibility with `checkTS.VGAMextra()`, while `ARMAXff()` is fitting an ARMA(2, 1):

```r
> fit.uschange3 <- vglm(Consumption ~ Income,

   ARMAXff(order = c(1, 2), noChecks = FALSE,

      zero = c("Var", "ARcoeff", "MAcoeff"),

      data = uschange)

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.
> coef(fit.uschange3, matrix = TRUE)

   drift.mean loglink(noiseVar) ARcoeff1 MAcoeff1 MAcoeff2
(Intercept) 0.1268513 -1.167491 0.6157928 -0.4457067 0.08024899
Income 0.2265722 0.000000 0.0000000 0.0000000 0.00000000

Internally, the roots of \( \vartheta(z) = 0 \) and \( \phi(z) = 0 \) over the estimated coefficients of the AR and MA components \( \vartheta \approx 0.616, \phi \approx (-0.446, 0.08)^T \) has been computed but not displayed. However, the prompted message indicates that all the roots (in module) lie outside the unit circle. The resulting roots are indeed shown along the VGLTSM–summary output, as well as the estimated linear predictor for \( \sigma^2_{\varepsilon_t} \) and the modified standard errors (Section 3.5), as follows:

```r
> summary(fit.uschange3)

Call:
`vglm(formula = Consumption ~ Income, family = ARMAXff(order = c(1, 2),
noChecks = FALSE, zero = c("Var", "ARcoeff", "MAcoeff")),
data = uschange)`

Pearson residuals:
   Min 1Q Median 3Q Max
drift.mean -8.6772 -0.3540 -0.007067 0.3621 12.091
loglink(noiseVar) -0.7068 -0.6484 -0.500030 0.0660 11.077
ARcoeff1 -8.0943 -0.2862 -0.013114 0.2480 1.374
MAcoeff1 -3.6296 -0.1094 0.005683 0.1119 6.479
MAcoeff2 -3.6296 -0.1031 0.005193 0.1261 12.057

Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) |
|----------|------------|---------|----------|
| (Intercept):1 | 0.12685 | 0.08618 | 1.472 0.141020 |
| (Intercept):2 | -1.16749 | 0.10337 | -11.295 < 2e-16 *** |
| (Intercept):3 | 0.61579 | 0.09427 | 6.532 6.49e-11 *** |
| (Intercept):4 | -0.44571 | 0.11916 | -3.740 0.000184 *** |
| (Intercept):5 | 0.08025 | 0.07327 | 1.095 0.273435 |
| Income | 0.22657 | 0.04517 | 5.016 5.29e-07 *** |
4.4. Supporting functions in VGAMextra

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 . ‘.’ 0.1 ‘ ’ 1

Number of linear predictors:  5

Names of linear predictors:
drift.mean, loglink(noiseVar), ARcoeff1, MAcoeff1, MAcoeff2

Log-likelihood: -156.1811 on 929 degrees of freedom

Number of iterations: 21

---

** Standard errors based on the asymptotic distribution of the MLE estimates:

<table>
<thead>
<tr>
<th>ARcoeff1</th>
<th>MAcoeff1</th>
<th>MAcoeff2</th>
<th>drift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.616</td>
<td>-0.446</td>
<td>0.080</td>
<td>0.289</td>
</tr>
<tr>
<td>s.e.</td>
<td>0.182</td>
<td>0.195</td>
<td>0.084</td>
</tr>
<tr>
<td></td>
<td>0.016</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Estimated linear predictor of sigma^2 (SD errors):

<table>
<thead>
<tr>
<th>(Intercept)</th>
<th>Income</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.167</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Loglikelihood: -156.181
AIC: 322.362, AICc: 322.694, BIC: 338.518

---

** Summary of checks on stationarity / invertibility:

Polynomial roots of the AR component computed from the estimated coefficients: (Examining stationarity/invertibility)

<table>
<thead>
<tr>
<th>Model1</th>
<th>Root1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.62338</td>
</tr>
</tbody>
</table>

Polynomial roots of the MA component computed from the estimated coefficients: (Examining stationarity/invertibility)

<table>
<thead>
<tr>
<th>Model1</th>
<th>Root1</th>
<th>Root2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.53553</td>
<td>3.53553</td>
</tr>
</tbody>
</table>

It is noted that all other VGLTSM family functions from the class VGLM–ARIMAX (such as ARXff()) operate in a similar fashion as ARMAXff() above. At present, ARMAX.GARCHff(), the VGLTSMff pertaining the class VGLM–ARMAX–GARCH, is endowed with capabilities to inspect the roots within the vglm() call, however, suitable summary methods are still under development, and will be incorporated over time.
Table 4.5.1. Miscellaneous functions in VGAMextra that assist VGLTSMffs in IRLS–Fisher scoring. Some of them are imported from other packages (details at the NAMESPACE).

<table>
<thead>
<tr>
<th>Function</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>dARp(), dMAq()</td>
<td>Compute the VGLTSM log–likelihood for VGLTSMffs from VGLM–ARIMAX and VGLM–ARMAX–GARCH.</td>
</tr>
<tr>
<td>dARMA()</td>
<td></td>
</tr>
<tr>
<td>stats::dpois()</td>
<td>Compute the VGLTSM log–likelihood for VGLTSMffs from VGLM–INGARCH.</td>
</tr>
<tr>
<td>stats::dnbinom()</td>
<td></td>
</tr>
<tr>
<td>VGAM::dlog()</td>
<td></td>
</tr>
<tr>
<td>VGAM::dyules</td>
<td></td>
</tr>
<tr>
<td>stats::ppois()</td>
<td>Density functions assisting PIT() (Section (4.4.1))</td>
</tr>
<tr>
<td>stats::pnbinom()</td>
<td>upon the predictive distribution $P(\cdot)$.</td>
</tr>
<tr>
<td>VGAM::plog()</td>
<td></td>
</tr>
<tr>
<td>VGAM::pyules</td>
<td></td>
</tr>
<tr>
<td>dmultinorm()</td>
<td>Compute the VGLTSM log–likelihood for VARff().</td>
</tr>
<tr>
<td>stats::dnorm()</td>
<td>Compute the VGLTSM log–likelihood for ARIMAX.errors.ff().</td>
</tr>
<tr>
<td>stats::dt()</td>
<td>Compute the VGLTSM log–likelihood for ARMA.studentt.ff().</td>
</tr>
<tr>
<td>ARpEIM.G2()</td>
<td>Compute the exact EIMs (returned in the form of the weight matrices) for VGLTSMffs from VGLM–ARIMAX and VGLM–ARMAX–GARCH.</td>
</tr>
<tr>
<td>MAqEIM.G2()</td>
<td></td>
</tr>
<tr>
<td>ARMA.EIM.G2</td>
<td></td>
</tr>
<tr>
<td>poissonTSff()</td>
<td>Dispatch the appropriate slots for VGLM.INGARCHff()</td>
</tr>
<tr>
<td>NegBinomTSff()</td>
<td>upon its argument dist.type.</td>
</tr>
<tr>
<td>yulesimonTSff()</td>
<td></td>
</tr>
<tr>
<td>logarithmicTSff()</td>
<td></td>
</tr>
</tbody>
</table>

4.5 Some miscellaneous functions

Table 4.5.1 presents some miscellaneous but relevant functions to assist VGLTSMffs while performing Fisher scoring, e.g., to compute the working weight matrices or the log–likelihood at each iteration. All these functions, with the exception of a few which are imported from VGAM, stats, or utils, have been implemented in VGAMextra and have been tested, although most of them remain undocumented. I will endeavour to complete this work later this year.

In addition, VGAMextra contains a set of utility functions required for secondary roles within VGLTSMffs. For instance, WN.lags() to compute lagged values of a given...
series; this is similar to `stats::embed()`, but conserving the initial dimension. The lagged positions are replenished with zeros by default. Type `?UtilitiesVGAMextra` in the R terminal to reveal the complete list of such.

### 4.6 Summary

This chapter is relevant for practitioners who wish to use VGLTSMs to model time series. It shows VGLTSMs handling different modelling structures using constraint matrices, by simultaneously introducing several ways to do so. The logical arguments `parallel` and `exchangeable` also provide convenient short-cuts for enforcing specific type of constraints while fitting VGLMs, but are yet not operational in VGLTSMffs and will be incorporated over time where they could be useful. See Section 3.3.1 as well as Chapter 8 in Yee (2015) for fuller details on both.

The results from this chapter also show how VGLTSMs are able to examine the important properties of stationarity, stability and invertibility for time series analysis, which endow TS models with parameters and structure that are consistent in time. Stability conditions are a result of difference equations theory that have conferred important properties to econometric time series. Invertibility has shown to play relevant roles in different areas of time series analysis, for instance, it underlies distributional MLE properties, and meaningful results derived from causality theory in economics (Brockwell and Davis (1991); Boudjellaba et al. (1992)). It is also relevant in a forecasting context, e.g., to allow innovations to be estimated from past observations (Box and Jenkins, 1970, Chapter 5).

However, VGLTSMs are broader and capable to cope with other multivariate TS models, particularly for cointegrated time series. This is the subject of the following chapter.
Chapter 4. On Supplementary Modelling Features of VGLTSMs
Chapter 5

VGLTSMs Towards Cointegrated Time Series

5.1 Introduction

Multivariate time series (MTS) are amenable to VGLTSMs by virtue of VGLMs which possess the ability to handle multiple responses, and this work has shown a first result in this line, presented in Section 3.4.1: the family function \texttt{VARXff} for VAR and VARX models. This chapter gives another example of the VGLTSM’s scope towards MTS by addressing systems of CTSs. The approach adopted here conforms with Engle and Granger (1987) via error correction models (ECMs) to detect and model accordingly stable long-run relationships between difference-stationary time series. At present, this subject has been developed for the bivariate case using VGLTSMs, giving place to another VGLTSM family function in \texttt{VGAMextra: ECM.EngleGran()}

To start with, it is necessary to shortly review the case of spurious regression by reason of its involvement with the ECM–framework. Much more comprehensive overviews of this topic may be found in the literature, e.g., Granger and Newbold (1974); Lee et al. (2005); Sun (2004); Ventosa-Santaulária (2009).

5.1.1 Spurious regression

In brief, spurious regression concerns the effect of stochastically unbalanced data on the results of regression analysis. It is a matter of series from independent DGPs but with strong temporal properties that may be found to incidentally conform with the OLS–regression/inference philosophy. Common examples are time series with a different trending mechanism, such as difference–stationary series.

Consider the classical linear model for time series regression with response $y_t$ and
a $K$–dimensional vector of predictors (exogenous) variables, $\mathbf{x}_t = (x_{t,1}, \ldots, x_{t,K})$, say:

$$y_t = \beta^T \mathbf{x}_t + \varepsilon_t, \quad \text{for } t = 1, \ldots, T,$$

(5.1.1)

where $\varepsilon_t$ is white noise. Fundamentally, three OLS–based inference scenarios for (5.1.1) can be outlined, as follows:

1. The processes $\mathbf{X}_t = (y_t, \mathbf{x}_t)^T$ are stationary. Hence the errors very likely will resemble white noise producing estimated coefficients $\beta$ generally agreeing with the properties of the OLS estimates. Ideally, the fixed–mean property of $y_t$ will render undistorted goodness of fit measures.

2. In the presence of trend–deterministic predictors, $x_{t,K}$, the OLS–inference assumptions are not generally violated when included in the regression analysis. Overall, there is no particular need of linearizing or detrending, as long as the responses are well–portrayed by (5.1.1).

3. The risk of spurious regression emerges when pairing predictors and responses that trend over time, such as difference stationarity or long–memory data, which will often lead to highly correlated errors producing misrepresented classical hypothesis tests on the coefficients. This fact is underlined by erroneous decisions from goodness of fit measures, such as the $R^2$, and inconsistent standard errors.

Statistically, one may escape from the effects of “stochastically unbalanced” regression by simply incorporating the differences of $\mathbf{X}_t = (y_t, \mathbf{x}_t)^T$ in the regression equation (5.1.1) but at the cost of other economic–theory related issues, see, e.g., Pfaff (2011, Chapter 4) for a short discussion on this.

5.2 Cointegration and ECMs

The previous section alludes to a number of potential drawbacks that may derive from involving trend–stochastic processes in a multivariate linear regression context, with spurious regression as a potential consequence. In this section, we address the special case of multivariate time series regression involving integrated time series of order $d$, say $\mathbf{y}_t = (y_{t,1}, y_{t,2}, \ldots, y_{t,K})$, which fundamentally concerns to unit root non–stationary processes, as defined by Engle and Granger (1987).

Definition 5.2.1. A univariate series, $y_t$, with no deterministic component that has a stationary, invertible ARMA representation after differencing $d$ times is said to be (process) integrated of order $d$, denoted as $y_t \sim I(d)$.

Ideally, one may wish to avoid spurious results such that the primary interest centers on identifying stable long–run relationships among the components of $\mathbf{y}_t$ or, more
specifically, to find modelling meanings and suitable estimation techniques to detect the presence of short–run or long–run stochastic disequilibrium and somehow correct this. A famed approach to address this matter was developed in the 1980s by Granger (1981) and Engle and Granger (1987) who proposed a two–step procedure involving error correction mechanisms upon a stationary linear combination of the series \{y_t, \ldots, y_{t,K}\} assumed as \(I(d)\)–variables, viz. the definition of cointegration.

**Definition 5.2.2.** (Engle and Granger, 1987) The components of a vector \(y_t\) are said to be cointegrated of order \(d, b\), if (a) all components of \(y_t\) are \(I(d)\), and (b) a non–zero vector \(\vartheta\) exists such that \(z_t = \vartheta^T y_t \sim I(d - b)\), \(b > 0, d \geq b\). This is denoted \(y_t \sim CI(d, b)\). The vector \(\vartheta\) is called a cointegrating vector.

The first step is to verify that the \((d)\)–variables are cointegrated. For instance, consider \(d = 1\), i.e. \(\{y_{t,1}, \ldots, y_{t,K}\}\) are all \(I(1)\)–type. In practice, estimating coefficients \(\beta \in \mathbb{R}^K\) such that the residuals of the regression \(y_{t,j} = \beta^T y_{t,(-j)} + z_t\) have no unit roots should suffice. That is, \(\hat{z}_t = (1, -\hat{\beta})^T \cdot (y_{t,j}, y_{t,(-j)}) \sim I(0)\), i.e., \(d = b = 1\) in Definition 5.2.2, and hence an estimated cointegrating vector would be given by \(\hat{\vartheta} = (1, -\hat{\beta})^T\). Here \(y_{t,(-j)} = (y_{t,1}, \ldots, y_{t,j-1}, y_{t,j+1}, \ldots, y_{t,K})^T\).

Secondly, upon stationarity of \(\hat{z}_t\), Engle and Granger (1987) state that an order\((u, v)\)–ECM, also denoted ECM\((u, v)\), can be specific as a mean to model the long–run stochastic trend of the series. When \(K = 2\), so that \(y_t = (y_{t,1}, y_{t,2})^T\), such ECM finds its general form as follows (Equations 4.5a and 4.5b in Pfaff (2011)):

\[
\begin{align*}
\nabla y_{t,1} | \varphi_{t-1} &= \psi_0 + \gamma_1 \hat{z}_{t-1} + \sum_{i=1}^{u} \psi_{i,1} \nabla y_{t-i,2} + \sum_{j=1}^{v} \psi_{j,2} \nabla y_{t-j,1} + \epsilon_{t,1}, \\
\nabla y_{t,2} | \varphi_{t-1} &= \phi_0 + \gamma_2 \hat{z}_{t-1} + \sum_{i=1}^{u} \phi_{i,1} \nabla y_{t-i,1} + \sum_{j=1}^{v} \phi_{j,2} \nabla y_{t-j,2} + \epsilon_{t,2},
\end{align*}
\]

\(t = 1, \ldots, T\), where \(\epsilon_t = (\epsilon_{t,1}, \epsilon_{t,2})^T \sim i.i.d. MWN(0, \Sigma)\), with covariance matrix \(\Sigma\). In this context, ECMs represent a mechanism to estimate how fast the dependent series returns to equilibrium (or corrects the disequilibrium) upon changes of the other series.

The error correction term, \(\hat{z}_t\), in (5.2.1) allows to correct disequilibrium at each period \((t - 1, t)\), hence the coefficient \(\gamma_1\) is expected to be negative in sign. This would be an indication that a stable long run relationship is ongoing, and hence one series may help to forecast the other. Bivariates ECMs as in (5.2.1) allow its extension to higher dimensional systems of CTSs, \(K > 2\), in a similar fashion. This multivariate ECM–version involves vector–time series (still past information), and thus cointegration in a broader sense. This instance is addressed in Campbell and Perron (1991).
5.3 The Engle–Granger procedure: an implementation in VGAMextra

Systems of CTSs in the sense of Engle–Granger are amenable to VGLTSMs as a result of (i) the VGLTSM’s ability to handle multiple responses, and (ii) the multivariate dependency of systems of CTSs on deterministic (only past) information, which may be accommodated by the VGLTSM–loglikelihood for estimation purposes. For instance, note that (5.2.1) can be seen as a VGLTSM with two responses,

$$\nabla y_{t,1} | \Phi_{t-1}, \nabla y_{t,2} | \Phi_{t-1} = \nabla y_t,$$

after imposing a suitable bivariate (conditional on $\Phi_{t-1}$) distribution on $\epsilon_t$, provided $(y_{t,1}, y_{t,2})^T \sim CI(1, 1)$.

At present, VGLTSMs can handle bivariate systems of $CI(1, 1)$–cointegrated time series by assuming i.i.d errors following a bivariate normal distribution. The VGLTSM family function implemented accordingly and subject of this section is ECM.EngleGran(), available in VGAMextra, which is based on VGAMextra:MVNcov(), a family function that estimates the parameters of the general multivariate Normal. The suffix "EngleGran" refers to the scheme adopted in this implementation: the Engle–Granger two–step procedure described in Section 5.2.

Most generally, the statistical framework managed by ECM.EngleGran() is:

$$((\epsilon_{t,1}, \epsilon_{t,2})^T \overset{i.i.d.}{\sim} N_2(0_2, \Sigma), \quad (\nabla y_{t,1} | \Phi_{t-1} = \mu \nabla y_{t,1} + \epsilon_{t,1}, \quad \nabla y_{t,2} | \Phi_{t-1} = \mu \nabla y_{t,2} + \epsilon_{t,2},$$

$$\mu \nabla y_{t,1} = \psi_0 + \beta_1^T x_t | \Phi_{t-1} + \gamma_1 \tilde{z}_{t-1} + \sum_{i=1}^u \psi_{i,1} \nabla y_{t-i,2} + \sum_{j=1}^v \psi_{j,2} \nabla y_{t-j,1},$$

$$\mu \nabla y_{t,2} = \phi_0 + \beta_2^T x_t | \Phi_{t-1} + \gamma_2 \tilde{z}_{t-1} + \sum_{i=1}^u \phi_{i,1} \nabla y_{t-i,1} + \sum_{j=1}^v \phi_{j,2} \nabla y_{t-j,2},$$

with $\Sigma = \begin{pmatrix} \sigma^2_{\epsilon_{t,1}} & \sigma_{\epsilon_{t,1}, \epsilon_{t,2}} \\ \sigma_{\epsilon_{t,1}, \epsilon_{t,2}} & \sigma^2_{\epsilon_{t,2}} \end{pmatrix}$, $\sigma_{\epsilon_{t,1}, \epsilon_{t,2}} = \sigma_{\epsilon_{t,1}} \cdot \sigma_{\epsilon_{t,2}} \cdot \rho$, where $\rho$ denotes the correlation between $x_t$ and $y_t$, and $x_t | \Phi_{t-1}$ signifies additional deterministic information whose effect on the system is to be explored. It transpires that (5.3.1) is a new sub–class of VGLTSMs, called VGLM–ECM, which accommodates (5.2.1) as a special case.

ECM.EngleGran() has five linear predictors:

$$\eta = (\mu \nabla y_{t,1}, \mu \nabla y_{t,2}, \log \sigma^2_{\epsilon_{t,1}}, \log \sigma^2_{\epsilon_{t,2}}, \sigma_{\epsilon_{t,1}, \epsilon_{t,2}})^T.$$

Here, any (deterministic) explanatories $x_t | \Phi_{t-1}$ are embedded in the models $\mu \nabla y_{t,1},$ and
\( \mu y_t \) by default, which means that \( \log \sigma_{\epsilon_t, 1}^2 \), \( \log \sigma_{\epsilon_t, 2}^2 \), and \( \sigma_{\epsilon_t, 1}, \epsilon_t, 2 \) are intercept–only, although this may be altered through the argument \texttt{zero}, while \( \phi_{i, 1}, \phi_{j, 2}, \psi_{i, 1}, \psi_{j, 2} \) (with \( 1 \leq i \leq u, 1 \leq j \leq v \)) \( \beta_{1} \), and \( \beta_{2} \) are coefficients to be estimated.

### 5.3.1 Score vector and EIM

Because of the distributional assumption on the errors \( \varepsilon_t \) any bivariate process, \( y_t = (y_{t, 1}, y_{t, 2})^T \), conforming with (5.3.1) follows a conditional bivariate normal distribution with mean \( \mu_y = (\mu_{y_{1}}, \mu_{y_{2}})^T \) and covariance matrix \( \Sigma \). The parameter vector pertaining to (5.3.1), and hence the parameter vector handled by \texttt{ECM.EngleGran()} is

\[
\theta = (\mu_{y_{1}}, \mu_{y_{2}}, \sigma_{\epsilon_{1}, 1}, \sigma_{\epsilon_{2}, 1}, \sigma_{\epsilon_{1}, 2})^T.
\]  

It is 
\textit{conditional} since the density of \( y_t \) carries all (known) information from explanatories \( x_t \), through the parameters \( \theta_j \) in (5.3.3) by means of the linear predictors (5.3.2). Consequently, for the 2–dimensional time series \( y_t \), and given \( x_t \), \( t = 1, 2, \ldots, T \), the log–likelihood of (5.3.1), denoted here \( \ell_t(\theta; y_t, x_t) = \ell_t \), is given by

\[
\ell_t(\theta; y_t, x_t|\Phi_{t-1}) = - \frac{T}{2} \log 2\pi - \frac{1}{2} \log \left| \Sigma(\theta) \right| - \frac{1}{2} \left[ y_t - \mu_y(\theta) \right]^T \Sigma^{-1}(\theta) \left[ y_t - \mu_y(\theta) \right],
\]  

so that the expressions for the EIMs and score vector derive accordingly. The score vector results from the first derivative of (5.3.4) with respect to \( \theta_j, j = 1, \ldots, 5 \), while the \((k, l)\)th element of the Fisher information matrix is given by

\[
[I(\theta)]_{k, l} = \frac{1}{2} \text{tr} \left\{ \Sigma^{-1}(\theta) \cdot \frac{\partial \Sigma(\theta)}{\partial \theta_k} \cdot \Sigma^{-1}(\theta) \cdot \left( \frac{\partial \Sigma(\theta)}{\partial \theta_l} \right) \right\} + \frac{1}{2} \left[ \frac{\partial \mu_y}{\partial \theta_k} \right]^T \cdot \Sigma^{-1}(\theta) \cdot \left[ \frac{\partial \mu_y}{\partial \theta_l} \right],
\]

Both are implemented in \texttt{ECM.EngleGran()} and compute the score vector and the EIM of (5.3.1) at each Fisher scoring iteration.

### 5.3.2 ECM.EngleGran() functional details

This section connects the preceding theory with its software implementation by giving several details of \texttt{ECM.EngleGran()} and its usage involving a two dimensional vector of \( I(1) \)–cointegrated time series, say \( y_t = (y_{t, 1}, y_{t, 2})^T \).

Within the VGLM–framework, \texttt{ECM.EngleGran()} is an object of class "vglmff", 
Table 5.3.1. Arguments handled by ECM.EngleGran().

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ecm.order</td>
<td>Length–2 vector of positive integers. The order ((u, v)) of the ECM to be fitted, cf (5.3.1).</td>
</tr>
<tr>
<td>zero</td>
<td>Character string. What linear predictor are to be modelled as intercept only? Same as in Table 3.3.3.</td>
</tr>
<tr>
<td>resids.pattern</td>
<td>Character string. Chooses the regression scheme to compute the residuals (\hat{z}_t). Choices are &quot;intercept&quot;, &quot;trend&quot;, &quot;neither&quot;, and &quot;both&quot;, as shown in Table 5.3.2. Default is &quot;intercept&quot;, that is, an intercept–only model, with no trend.</td>
</tr>
<tr>
<td>lag.res</td>
<td>Positive integer. What lag of the residuals (\hat{z}_t) is to be included in the analysis? Default is 1, i.e., the previous period (t - 1).</td>
</tr>
<tr>
<td>lmean, lvar, lcov</td>
<td>Link functions applied to linear predictors: (\mu_{\nabla y_t, 2}, \mu_{\nabla y_t, 1}; \sigma_{\epsilon_t, 1}^2, \sigma_{\epsilon_t, 2}^2; ) and (\sigma_{\epsilon_t, 1}, \epsilon_{t, 2}) respectively.</td>
</tr>
<tr>
<td>ordtsDyn</td>
<td>Non–negative integer. Enables the ECM–modelling structure handled VECM() from package tsDyn (Section 5.3.2).</td>
</tr>
</tbody>
</table>

and, as such, it becomes operational by means of the modelling function vglm() with calls of the form:

```r
> ## A 'vglm()'-call using 'ECM.EngleGran()'
> vglm(formula, family = ECM.EngleGran(), data, ....)
> ## ECM.EngleGran() belongs to the "vgltsmff"--class
> class(ECM.EngleGran())
[1] "vglmff"
attr(,"package")
[1] "VGAM"
```

Internally, ECM.EngleGran() operates according to the following three stages at the "vglm()" call (in this order): First, it computes the equilibrium errors. For the bivariate case, these are the residuals

\[
\hat{z}_t = y_{t, 2} - \hat{\beta}_1 y_{t, 1}
\]  

(5.3.6)

from the OLS–regression \(y_{t, 2} \sim y_{t, 1}\). The (possibly cointegrating) vector \(\hat{\Theta} = (1, -\hat{\beta}_1)^T\), comprising the estimated OLS–coefficient \(\hat{\beta}_1\), is printed upon convergence of the Fisher scoring. However, several choices are available to carry out the regression (5.3.6), by means of the argument resids.pattern, allowing, e.g., deterministic trends, as shown in Table 5.3.2. The error term is introduced by \(z_t\).
Table 5.3.2. Regression schemes managed by ECM.EngleGran() to estimate residuals and cointegrating vectors using resids.pattern.

1) If resids.pattern = "intercept", then it fits \( y_{t,2} = \beta_0 + \beta_1 y_{t,1} + z_t \), and \( \hat{\vartheta} = (1, -\hat{\beta}_1, -\hat{\beta}_2)^T \) is printed.

2) If resids.pattern = "trend", then \( y_{t,2} = \beta_1 y_{t,1} + \beta_2 t + z_t \) is fitted, and \( \hat{\vartheta} = (1, -\hat{\beta}_1, -\hat{\beta}_2)^T \) is printed out.

3) If resids.pattern = "neither", then \( y_{t,2} = \beta_1 y_{t,1} + z_t \) is fitted, printing \( \hat{\vartheta} = (1, -\hat{\beta}_1)^T \).

4) If resids.pattern = "both", then \( y_{t,2} = \beta_0 + \beta_1 y_{t,1} + \beta_2 t + z_t \), while \( \hat{\vartheta} = (1, -\hat{\beta}_0, -\hat{\beta}_1, -\hat{\beta}_2)^T \) is printed out.

From (5.3.6), note that \( y_{t,2} \) is assumed to be the response and \( y_{t,1} \) the regressand in this stage. While the inputting order seems not to convey any loss of generality over such regression (one can rearrange the entries in the vector \( y_t \)), it will play a relevant role when entering the formula in the "vglm()" call.

Secondly, ECM.EngleGran() tests whether the series \( y_t = (y_{t,1}, y_{t,2})^T \) are cointegrated. More precisely, in this step ECM.EngleGran() analyses the presence of unit roots in the residual series \( \{\hat{z}_t\} \). Internally, this occurs in @initialize assisted by the function KPSS.test() (Section 4.4.2), where the null hypothesis of level or trend stationarity on the residuals, \( z_t \), computed as per Table 5.3.2 is tested at 5% level. A warning message is displayed accordingly after convergence of the Fisher scoring. For an inclusive test, one can use VGAMextra::KPSS.test() on the residuals, which can be retrieved from fit.ECM with the call fit.ECM@misc$coint.res.

Note that potential cointegrating vectors, \( \hat{\vartheta}^T \), can be identified at this stage. For instance, resids.pattern = "neither" implies residuals of the form
\[
\hat{z}_t = y_{t,2} - (\hat{\beta}_1 \cdot y_{t,1}) = (1, -\hat{\beta}_1) \cdot (y_{t,2}, y_{t,1})^T = \hat{\vartheta}^T y_t,
\]
with \( y_{t,i} \sim I(1), i = 1, 2 \). If the series \( \{z_i\} \) occurs to be \( I(0) \)-integrated, that is, unit roots free, then the estimated \( \hat{\vartheta} \), as shown in Table 5.3.2, becomes a cointegrating vector for the components of \( y_t \).

Lastly, Engle.EngleGran() estimates an order–(u, v) ECM as in (5.2.1) assuming zero–mean bivariate normal errors, \( \varepsilon_t = (\varepsilon_{t,1}, \varepsilon_{t,2})^T \), as a special case of (5.3.1). Most importantly, at the vglm() call, this VGLTSMff will fit such model to \( y_t \), regardless of whether the series \( y_{t,1} \) and \( y_{t,2} \) are cointegrated or not. Followed from Engle and


Granger (1987), special care is needed here to detect spurious results when \( y_{t,1}, y_{t,2} \) fail to meet this property.

A few additional notes:

1. Due to the stationarity of the residuals \( \hat{z}_t \), any lagged value \( \hat{z}_{t-\iota}, \iota > 1 \) rather than just the residuals from the previous period \( \hat{z}_{t-1} \) may be embedded in (5.3.1). ECM.EngleGran() accommodates this change through the argument lag.res, whose default is lag.res = 1, hence \( \hat{z}_t \).

2. Since they will be internally differenced, the undifferenced series, \( y_{t,1}, y_{t,2} \), must be entered as a bivariate response using cbind() in the formula, even when explanatories \( \mathbf{x}_t|\Phi_{t-1} \) (\( x_2t \) and \( x_3t \) in the code below) are to be incorporated in the analysis. Thus, fitting calls should be similar to the following:

```r
### The data, \((y_{1t}, y_{2t})\) is the bivariate response.
coint.data <- data.frame(y1 = y1t, y2 = y2t, x2 = x2t, x3 = x3t)

# Intercept--only ECM model.
vglm(cbind(y1, y2) ~ 1, ECM.EngleGran(ecm.order), data, ...)

# Including covariates x2 and x3 (not differenced) and
# are used in the conditional mean models (default), i.e.,
# zero = c("var", "cov")
vglm(cbind(y1, y2) ~ x2 + x3, ECM.EngleGran(ecm.order),
    data = coint.data,...)
```

Similar to other VGLTSMs, the above call will create an object of class "vglm", or "vglm"-object. Note that no arguments are available to internally difference \( x_2t \) or \( x_3t \), such that, if applicable, covariates must be differenced prior the model fitting and then incorporated in the formula, e.g.,

```r
### A data set storing the first diff. of x2 and x3.
coint.data.diff <- transform(coint.data, diffx2 = diff(x2),
                             diffx3 = diff(x3))
vglm(cbind(y1, y2) ~ diffx2 + diffx3, ECM.EngleGran(ecm.order),
     data = coint.data.diff,...)
```

3. The position of the series \( y_{t,1} \) and \( y_{t,2} \) in the inputted bivariate response, \( \text{cbind}(y_1, y_2) \), at the vglm() call is of utmost importance. Internally, the residuals, as depicted in Table 5.3.2, are by default computed from the regression Column 2 \( \sim \) Column 1, or \( y_2 \sim y_1 \) in this case. This order cannot be modified.
5.3. The Engle–Granger procedure: an implementation in VGAMextra

4. By default, \( \text{zero} = ("\text{var}", \"\text{cov}\") \), that is, the linear predictors involving elements of the covariance matrix \( \Sigma \) are modelled as intercept–only, viz. \( \log \sigma_{\varepsilon_{t,1}}^2 \), \( \log \sigma_{\varepsilon_{t,2}}^2 \), and \( \sigma_{\varepsilon_{t,1}, \varepsilon_{t,2}} \). Analogous to other VGLTSMff, this choice may be over-ridden, if required.

The choices accepted by the argument "\text{zero}" are: "\text{var}", "\text{cov}", and "\text{Diff}". Here, "\text{Diff}" stands for Diff'erenced conditional means. Thus, e.g., \( \text{zero} = \"\text{Diff}\" \) will cause the conditional mean linear predictors, \( \mu_{\nabla y_{t,2}}, \mu_{\nabla y_{t,1}} \), to be modelled as intercept–only.

5. At present, no summary or forecasting methods have been implemented for "\text{vglm}"–objects created with \texttt{ECM.EngleGran()}. This is part of the future work in this area.

5.3.3 A simulated example

Using artificial data this section shows \texttt{ECM.EngleGran()} applied to two \( I(1) \)–cointegrated time series via the Engle–Granger scheme, as described in Section 5.3.2.

It is noted that \texttt{VECM()} from the package tsDyn (Fabio Di Narzo et al., 2009) is also equipped with functions to address systems of CTSs approached by ECMs in a similar fashion as \texttt{ECM.EngleGran()}, however, the bivariate–ECM handled in \texttt{VECM()} slightly differs from our approach as it considers the current residuals, \( \hat{z}_t \), instead of \( \hat{z}_{t-\kappa}, \kappa \geq 1 \), in the conditional mean models \( \mu_{\nabla y_{t,1}} \) and \( \mu_{\nabla y_{t,2}} \). Yet, \texttt{ECM.EngleGran()} is able to accommodate such modelling structure via the argument \texttt{ordtsDyn}. This will be illustrated later in this section.

5.3.3.1 Generating two \( I(1) \)–integrated series

We begin by generating two \( I(1) \)–series \( (n = 100) \) but cointegrated of order \( (1,1) \). For simplicity, these are two random walks denoted \( \mathbf{y}_t = (x_t, y_t)^T \) with the following structure:

\[
\begin{align*}
  x_t &= x_{t-1} + \varepsilon_{t,1}, \\
  y_t &= \beta_0 + \beta_1 x_t + \varepsilon_{t,2},
\end{align*}
\]  
(5.3.7)

where \( \varepsilon_t = (\varepsilon_{t,1}, \varepsilon_{t,2})^T \sim N_2(0, \Sigma), \Sigma = \begin{pmatrix} \sigma_{\varepsilon_{t,1}}^2 & \sigma_{\varepsilon_{t,1}} \sigma_{\varepsilon_{t,2}} \rho \\ \sigma_{\varepsilon_{t,1}} \sigma_{\varepsilon_{t,2}} \rho & \sigma_{\varepsilon_{t,2}}^2 \end{pmatrix} \), with, e.g.,

\[
\sigma_{\varepsilon_{t,1}} = \exp(2 \cdot \log(1.5)), \quad \sigma_{\varepsilon_{t,2}} = \exp(2 \cdot \log(0.75)), \quad \rho = 0.75, \quad (\beta_0, \beta_1)^T = (0.0, 0.60)^T.
\]

The code utilized is given below, whereas the plot of both simulated series is shown in Figure 5.3.1. The errors \( \varepsilon_t \) are i.i.d. and randomly generated with \texttt{VGAM::rbinorm()}.
which generates random vectors from the bivariate normal distribution.

Modelling the dynamic behavior of (5.3.7) according to Engle and Granger (1987) heavily relies on the assumption that $x_t$ and $y_t$ are cointegrated, or else run the risk of spurious results. A rather convenient way to verify this follows from Section 5.3.2 focussing on the residuals $\hat{z}_t$ of the regression $y_t \sim x_t$ (with no trend–term), which are ideally expected to conform with stationary conditions. In particular, one can find up to three choices in VGAMextra to formally test this, as follows:

(i) Using `KPSS.test()`, which explores the existence of unit roots of a given series (Section 4.4.2). For our data $y_t = (x_t, y_t)^T$, a level–stationarity test on such residuals should conform with the generating process (5.3.7).
5.3. The Engle–Granger procedure: an implementation in VGAMextra

(ii) By fitting an AR(1) model with no drift over the residual series \( \{ \hat{z}_t \} \), say \( \hat{z}_t = \theta_1 \hat{z}_{t-1} + w_t \), \( \{ w_t \} \) signifying white noise, and then verify that the root \( \alpha \) of the equation \( 1 - \hat{\theta}_1 \alpha = 0 \) lies outside the unit circle.

Both, (i) and (ii) require the residuals to be computed beforehand.

(iii) Using \texttt{ECM.EngleGran()} which internally computes the residuals upon the argument \texttt{resids.pattern}, followed by testing stationarity on such using \texttt{KPSS.test()}.

Before addressing (i)–(iii), let us first estimate by OLS the residuals \( \hat{z}_t \) where the regression model considered will be \( y_t = \beta_0 + \beta_1 x_t + z_t \), due to the generation process in (5.3.7). The plot of the resulting residuals, \( \{ \hat{z}_t \} \) is shown in Figure 5.3.2.

> # Residuals from the static regression \( y_t \sim x_t \), with intercept.
> errors.coint <- residuals(lm(yt ~ xt))

As proposed in (i), we evaluate \( \{ \hat{z}_t \} \) at \texttt{VGAMextra::KPSS.test()} with \texttt{type.H0 = "level"}, whose output is shown next. Here, the null hypothesis states that residuals \( \hat{z}_t \) conform with stationary conditions, in the sense of “unit root” free.

> ### Function KPSS.test applied to the residuals \( \{ \hat{z}_t \} \)
> kpss.VGAMextra <- KPSS.test(x = errors.coint, type.H0 = "level")

\begin{verbatim}
H0: level stationary vs. H1: Unit root.
Test statistic: 0.09517007
p-value: 0.3378543
Upper tail percentiles:
10% 5% 2.5% 1%
Critical value 0.347 0.463 0.573 0.739
\end{verbatim}
The AR(1) model suggested in (ii) can be readily fitted with VGAM extra: ARXff() making use of nodrift = TRUE and noChecks = FALSE. Recall that noChecks = FALSE internally enables checkTS.VGAMextra() which computes the polynomial roots requested. The code is the following:

```r
> data.errors <- data.frame(est.errors = errors.coint)
> fit.root.test <- vglm(est.errors ~ 1,
  family = ARXff(order = 1, nodrift = TRUE,
  noChecks = FALSE),
  data = data.errors, trace = FALSE)
```

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.

Both results suggest that \( x_t \) and \( y_t \) are cointegrated, i.e., \( CI(1, 1) \), as no unit roots for \( \{\hat{z}_t\} \) have been detected from (i)–(ii). Alternatively, ECM.EngleGran() may be directly used for this purpose, nonetheless postponed for the next stage.

### 5.3.3.2 The Engle–Granger procedure using ECM.EngleGran()

Once the presence of unit roots has been rejected for \( \{\hat{z}_t\} \), we proceed to the next stage, such that and ECM–like model as in (5.2.1) can be specified to describe the dynamic behaviour of \( y_t = (x_t, y_t)^T \) for certain order \((u, v)\). Assuming zero–mean bivariate normal errors, such ECM\((u, v)\) can be regarded as a VGLTSM with vector response given by \( (\nabla x_t, \nabla y_t)^T \sim N_2(\mu_{\nabla x_t}, \mu_{\nabla y_t})^T, \Sigma) \),

\[
\begin{align*}
  \mu_{\nabla x_t} &= \phi_0 + \gamma_1 \hat{z}_{t-1} + \sum_{i=1}^u \phi_{i,1} \nabla y_{t-i} + \sum_{j=1}^v \phi_{j,2} \nabla x_{t-j}, \\
  \mu_{\nabla y_t} &= \psi_0 + \gamma_2 \hat{z}_{t-1} + \sum_{i=1}^u \psi_{i,1} \nabla x_{t-i} + \sum_{j=1}^v \psi_{j,2} \nabla y_{t-j},
\end{align*}
\]

with \( \Sigma \) as in (5.3.1). Indeed, (5.3.8) pertain to the class–(5.3.1), and hence can be handled by ECM.EngleGran().

In this regard and merely based on the data generating process (5.3.7), we will estimate a VGLM–ECM of order–(1, 1), or simply ECM(1, 1), with no covariates \( x_t | \Phi_{t-1} \) over the generated data \( y_t = (x_t, y_t)^T \). Observe that this modelling framework coincides with (5.3.8) for \( u = 1 \) and \( v = 1 \). The following code chunk executes one choice of R code for this, followed by the subsequent output. The generated "vglm"–
5.3. The Engle–Granger procedure: an implementation in VGAMextra

class object, that is, the VGLM–ECM(1, 1) fitted, saved as `fit1.coint.VGAMextra`:

```r
> coint.Data <- data.frame(xt = xt, yt = yt)
> fit1.coint.VGAMextra <- vglm(cbind(xt, yt) ~ 1,
ECM.EngleGran(ecm.order = c(1, 1),
  resids.pattern = "intercept",
  zero = c("var", "cov")),
  trace = TRUE, data = coint.Data)
```

```
VGLM linear loop 1 : loglikelihood = -295.84016
VGLM linear loop 2 : loglikelihood = -274.58852
VGLM linear loop 3 : loglikelihood = -274.2828
VGLM linear loop 4 : loglikelihood = -274.28277
VGLM linear loop 5 : loglikelihood = -274.28277
```

The residuals appear to be free of unit roots.

Estimated co-integrating vector:

```
  betaY2 (Intercept) betaY1
1.0000000 -0.5377030 -0.6410308
```

Final sample size: 98

Here are a few comments.

1. The estimated cointegrating vector, is \( \hat{\Theta} = (1.00, 0.538, 0.64)^T \), cf. **Definition 5.2.2**, which is internally estimated by `ECM.EngleGran()` and printed after the Fisher scoring has converged.

   The residuals have been computed from the model \( y_t = \beta_0 + \beta_1 \cdot x_t + z_t \), because `resids.pattern = "intercept"`.

2. The estimated coefficients can be retrieved from the object `fit1.coint.VGAMextra`, producing the next output:

```r
> coef(fit1.coint.VGAMextra, matrix = TRUE)
```

```
  Diff1  Diff2 loglink(var1) loglink(var2) cov12
(Intercept)  0.108140 0.057127  0.8733 0.9979 2.3604
ErrorsLag1  0.117404 -0.978652  0.0000  0.0000  0.0000
diffy1Lag1 -0.099274  0.620915  0.0000  0.0000  0.0000
diffy2Lag1  0.158701 -0.937915  0.0000  0.0000  0.0000
```

The estimated \( \hat{\gamma}_1 \approx -0.979 \) is negative and close to unity in absolute value, as expected, assuring the system convergence to its long-run equilibrium path. Overall, results show that \( y_{1,t} \) and \( y_{2,t} \) are two cointegrated \( I(1) \)-variables guaranteeing Granger causality (Engle and Granger, 1987) in one direction whereby at least one series can help to forecast the other.
5.3.3.3 The Engle–Granger procedure using \texttt{tsDyn::VECM()}

It was previously mentioned that \texttt{tsDyn} also possesses infrastructure to handle ECMs, but marginally different from (5.3.8) as it embeds $\hat{z}_t$ instead of $\hat{z}_{t-\kappa}, \kappa > 0$ in the models $\mu \nabla y_{t,1}$ and $\mu \nabla y_{t,2}$. This modelling framework however is also amenable to the VGLTSMS log–likelihood, and particularly handled by \texttt{ECM.EngleGran()}, where the equilibrium error $\hat{z}_t$ is allocated at $x_t|\Phi_{t-1}$. In the following we compare the performance of \texttt{ECM.EngleGran()} and \texttt{tsDyn::VECM()}, after fitting similar ECMs to our simulated data. Note that a few modifications are needed on \texttt{ECM.EngleGran()}. The central arguments will be (i) \texttt{lag = 1} in \texttt{tsDyn::VECM()}, the counterpart of \texttt{ecm.order = c(1, 1)}, and (ii) \texttt{ordtsDyn = 1} in \texttt{ECM.EngleGran()} to accommodate the residuals accordingly and hence emulate the model handled by \texttt{tsDyn::VECM()}. The code is as follows. In particular, observe the prompted message from \texttt{ECM.EngleGran()} pointing the replica.

```r
> library("tsDyn")
> ## MODEL 1: Estimation using 'VECM()'
> fit.coint.tsDyn <- with(coint.Data, VECM(data = cbind(yt, xt), lag = 1, 
est = "2OLS")
> summary(fit.coint.tsDyn)

###Model VECM
Full sample size: 100 End sample size: 98
Number of variables: 2 Number of estimated slope parameters 8
AIC 22.05 BIC 45.315 SSR 509.78
Cointegrating vector (estimated by 20LS):
  yt   xt
r1  1 -0.68381

ECT Intercept yt -1
Equation yt -0.6251(0.3374). 0.0953(0.1742) -0.0826(0.2101)
Equation xt 0.2858(0.3094) 0.0976(0.1598) -0.0287(0.1927)
  xt -1
Equation yt -0.0260(0.1882)
Equation xt -0.0095(0.1726)
> fit2.coint.VGAMextra <-
  vglm(cbind(xt, yt) ~ 1, ECM.EngleGran(ecm.order = c(1, 1),
                                    resids.pattern = "neither",
```
5.3. The Engle–Granger procedure: an implementation in VGAMextra

```r
ordtsDyn = 1,
data = coint.Data, trace = FALSE)
```

The residuals appear to be free of unit roots.

Estimated co-integrating vector:

- `betaY2` betaY1
  - 1.00000 -0.68381

Final sample size: 98

Argument 'ordtsDyn' is on. Results should be similar to those from VECM(), from package 'tsDyn'.

```r
> ( coef.ECM.2 <- coef(fit2.coint.VGAMextra, matrix = TRUE) )
```

<table>
<thead>
<tr>
<th></th>
<th>Diff1</th>
<th>Diff2</th>
<th>loglink(var1)</th>
<th>loglink(var2)</th>
<th>cov12</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.0975549</td>
<td>0.095254</td>
<td>0.86556</td>
<td>1.0387</td>
<td>2.3816</td>
</tr>
<tr>
<td>ErrorsLag1</td>
<td>0.2857812</td>
<td>-0.625125</td>
<td>0.00000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>diffy1Lag1</td>
<td>-0.0095033</td>
<td>-0.025992</td>
<td>0.00000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>diffy2Lag1</td>
<td>-0.0286825</td>
<td>-0.082588</td>
<td>0.00000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>

> ###
> ### The estimated coefficients.
> ###

```r
> t(coef.ECM.2[, 1:2][c(2, 1, 4, 3), ][, 2:1])
## FROM 'VGAMextra'
```

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>ErrorsLag1 (Intercept) diffy2Lag1 diffy1Lag1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diff2</td>
<td>-0.62512</td>
<td>0.095254 -0.082588 -0.0259920</td>
</tr>
<tr>
<td>Diff1</td>
<td>0.28578</td>
<td>0.097555 -0.028683 -0.0095033</td>
</tr>
</tbody>
</table>

The output above shows `ECM.EngleGran()` coping with the rearranged–errors scenario, mimicking `VECM()`. The central step of computing such equilibrium errors is indeed similarly performed by both frameworks, as described in Table 5.3.2, being the argument `include`, in `VECM()`, the counterpart of `resids.pattern`.

5.3.3.4 An alternative based on `lm()`

A third option to address the Engle–Granger procedure is proposed in Pfaff (2011, Chapter 4), which provides snippets of `R` code for this based on simple regression with `stats::lm()`. The ECM structure handled here coincides with (5.2.1).
For the sake of comparison, we evaluate our simulated data on the referred R code, whose results are shown below, and stored in the object `ecm.reg`. Note that several computations must be manually effected beforehand, such as differences and lagged values, in addition to the fact that this methodology skips the initial verification of residuals stationarity, similar to `tsDyn::VECM()`. Recall that our data is denoted as `xt, yt`:

```
#### Routine from Pfaff (2011), Chapter 4 ####
## Compute and save the residuals form the regression yt ~ xt.
errors.coint <- residuals(lm(yt ~ xt)) # Residuals

## Compute Order--1 differences.
difx1 <- diff(ts(xt), lag = 1, differences = 1) # First difference for xt
dify1 <- diff(ts(yt), lag = 1, differences = 1) # First difference for yt

## Set up the dataset (coint.data), including Order-2 lagged differences.
coint.data <- data.frame(embed(difx1, 2), embed(dify1, 2))
colnames(coint.data) <- c("difx1", "difxLag1", "dify1", "difyLag1")

## Remove unutilized lagged errors accordingly.
errors.cointLag1 <- errors.coint[1:(n - warm.up - 2)]
coint.data <- transform(coint.data, errors.cointLag1 = errors.cointLag1)

## Use lm() to regress 'dify1' on 'errors.cointLag1' + 1 lagged differences.
## This call fits the ECM, similar to VECM() and ECM.EngleGran().
ecm.reg <- lm(dify1 ~ errors.cointLag1 + difxLag1 + difyLag1,
              data = coint.data)
```

Finally, we show the estimated coefficients compared to results from Section 5.3.3.2, using `ECM.EngleGran()`.

```
> ## The estimated coefficients.
> coefVGAMextra <- coef(fit1.coint.VGAMextra, matrix = TRUE)[, 2]
> data.frame(Pfaff2011 = coef(ecm.reg), VGAMextra = coefVGAMextra)

                Pfaff2011 VGAMextra
(Intercept)  0.057127  0.057127
errors.cointLag1 -0.978652 -0.978652
difxLag1       0.620915  0.620915
difyLag1      -0.937915 -0.937915
```

It is pointed out that estimates from `tsDyn:VECM()` are not comparable with those from Sections 5.3.3.2, and 5.3.3.4 because the ECM structures managed by the functions differ.
5.4 Summary

This chapter introduces ECM.EngleGran(), a VGLTSMff towards modelling the potential long-run equilibrium relevant to the components of a 2-dimensional vector of, possibly stochastically unbalanced, time series, and somehow identify spurious results. The (bivariate) error correction–type model handled by ECM.EngleGran() is shown in (5.3.1), with bivariate normal errors underneath. As a special case, ECM.EngleGran() accommodates ECM representations based on the Engle–Granger methodology (Engle and Granger, 1987), viz. (5.2.1).

Regarding its implementation in R, for a given bivariate TS vector \( y_t = (y_{t,1}, y_{t,2})^T \), ECM.EngleGran() estimates (i) the coefficients of the regression \( y_{t,2} \sim y_{t,1} \), i.e., a potential cointegrating vector provided stationary residuals, and (ii) a model like (5.3.1) with the five linear predictors, (5.3.2), where covariates of interest are included in the conditional mean models \( \mu_{\nabla y_{t,1}}, \mu_{\nabla y_{t,2}} \) by default. Further, this ECM.EngleGran() has been contrasted with two alternatives to estimate ECM–like models showing consistent results, viz. tsDyn::VECM() and a short code snippet provided in Pfaff (2011). This VGLTSMff has shown to confer modelling advantages over such frameworks, e.g., the ability to incorporate either covariates in the volatility model via the argument zero, or suitable VGLM–link functions over the linear predictors. VECM() also allows the inclusion of deterministic regressors but only over the long–term relationship (conditional mean) model. Also, the variance–covariance structure underlying the long–run relationship is estimated by ECM.EngleGran(). In our artificial example (5.3.7), these are \( \hat{\sigma}^2_{\varepsilon_{t,1}} \approx 2.39, \hat{\sigma}^2_{\varepsilon_{t,2}} \approx 2.71, \) and \( \hat{\sigma}_{\varepsilon_{t,1}, \varepsilon_{t,2}} \approx 2.36, \) retrieved from

```r
> coef(fit1.coint.VGAMextra, matrix = TRUE)

Diff1 Diff2 loglink(var1) loglink(var2) cov12
(Intercept)  0.108140  0.057127  0.8733  0.9979  2.3604
ErrorsLag1    0.117404 -0.978652  0.0000  0.0000  0.0000
diffy1Lag1    -0.099274  0.620915  0.0000  0.0000  0.0000
diffy2Lag1    0.158701 -0.937915  0.0000  0.0000  0.0000
```

Further enhancements are planned for ECM.EngleGran(), such as ECM–forecasting methods. This VGLTSMff supports bivariate time series only, and hence is restricted to explore \( R = 1 \) cointegrating relationships, unlike VECM() which handles \( R \geq 1 \). However, VECMs for multiple CTS are to be addressed using, e.g., VGAM::trinormal() for \( N_3 \) responses, and VGAMextra::MVNcov(). In terms of coping computationally towards estimating systems of CTS as \( u \) and \( v \) increase, the next chapter shows a strategy using ECM.EngleGran() via the class of RR–VGLMS, which is a class of models for dimension reduction.
Chapter 6

Applications of VGLTSMs to Real Data

6.1 Introduction

The previous chapters defined and characterized various classes of VGLTSMs, including many details on the R–implementation of VGLTSMs and its usage. In addition, several examples showing VGLTSMs accommodating typical models for time series with emphasis on modelling and estimation have been examined, but mostly involving artificial data. In this chapter VGLTSMs are applied to some real–life datasets (except Section 6.4), where results are compared to other modelling frameworks and R packages for time series analysis. A number of VGLTSM capabilities are to be illustrated such as the analysis of time series of counts. Section 6.4 particularly focuses on how RR–VGLMs may help to aid systems of CTS estimation with simulated data.

6.2 Parameter constraints

We first analyze the variability of particulate matter levels smaller than 10µm (PM$_{10}$) in Mexico City, between January 2004 and June 2005. This and other variables, e.g., meteorological, were retrieved from the Atmospheric Monitoring System, Mexico City, and are available in VGAMextra as ap.mx. Table 6.2.1 gives a description of such.

Following standard modelling practice and supported by a vast literature, an initial approach towards this type of data may be to explore the effects of air pollutants exposure on hospital admissions due to respiratory symptoms, or over subpopulations with an elevated risk of dying due to poor health conditions (Pope and Schwartz, 1996; Schwartz, 2000; Chit-Ming et al., 2008; Romieu et al., 2012; Blanco-Becerra et al.,
Table 6.2.1. Variables included in \textit{ap.mx}, collected 1\textsuperscript{st} January 2004–30\textsuperscript{th} June 2005.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>PM10</td>
<td>24–hr average\footnote{†} concentration of particulate matter smaller than 10 µm (PM$_{10}$)</td>
<td>Micrograms per milliliter</td>
</tr>
<tr>
<td>O3</td>
<td>Daily maximum 8–hour moving average of Ozone\footnote{‡} (O$_3$)</td>
<td>Micrograms per milliliter</td>
</tr>
<tr>
<td>Temp</td>
<td>Daily mean average of temperature (Temp)</td>
<td>Degrees celsius</td>
</tr>
<tr>
<td>HR</td>
<td>Daily mean average of relative humidity (HR)</td>
<td>Percentage (%)</td>
</tr>
</tbody>
</table>

\footnote{†}{From 00:00 to 23:59 hrs.}
\footnote{‡}{This is the maximum over all the “sliding” 8–hour–windows, from 00:00 to 23:59 hrs.}

...2014). In such cases, the sub–class VGLM–INGARCH offers a broad modelling framework for time series of counts, whose performance with real data is to be demonstrated in Section 6.3.

Alternatively, a more elaborated purpose, potentially attractive to contingency–plans implementations, may relate to predict future emissions of PM$_{10}$ or O$_3$ accounting for a heteroskedastic structure on the forecast variance and some seasonal effects on the conditional mean model. Here, the GARCH–class appears to be a suitable choice. However, besides an ordinary GARCH structure, one may be primarily interested on identifying external factors intervening with the series volatility in such a way that the forecast variance may indeed conform with a process driven by separate time–varying processes, for instance, meteorological parameters. This modelling structure is not amenable to the GARCH framework but it is readily accommodated by the VGLM–ARIMAX($p$, $d$, $q$) class through the family function \texttt{ARMAXff()} due to its ability to constrain the effect of covariates towards the linear predictor managing the forecast variance, and still accounting for an ARMA structure on the conditional mean model. This analysis represents our specific goal in this section.

While the results from examining any of both pollutants may be of interest, for instance in environmental research, we focus only on PM$_{10}$ for brevity, taking O$_3$ and meteorological variables as covariates. Figure 6.2.1 shows the time plot, PACF, and ACF respectively of PM$_{10}$, whilst Figure 6.2.2 shows trend–scatter plots of this pollutant against up to order–3 lagged series of O$_3$. Note that some central features concerning the forecast variance, denoted, $\sigma^2_{\epsilon|PM_{10}\Phi_{t-1}}$, are suggested from the scatter plots. The ACF and the time plot suggest the non–stationarity of the PM$_{10}$ series: the ACF seems to ‘decay’ slowly from lag–2, yet remaining well above the threshold range. However, this section is unconcerned about \textit{differencing}, where instead we
6.2. Parameter constraints

aim to empirically model heteroskedasticity. Also, Figure 6.2.2 (a)–(c) gives moderate graphical-evidence of O₃ levels intervening with the PM₁₀ volatility: The data somewhat disperses at a higher rate for lagged values of O₃.

Other features may be noted due to its potential influence on the expected daily levels of PM₁₀, denoted here as \( \mu_{PM_{10},t|\Phi_{t-1}} \). For instance, the PACF (Figure 6.2.1) and the scatter plots (d)–(f) from Figure 6.2.2 somewhat suggest an order-1 autoregressive term, while the ACF (Figure 6.2.1) seemingly drops (yet above the threshold) from lag-2, recommending a seasonal MA(2) term. Based on this, the following statistical structure is proposed to investigate the referred PM₁₀ levels:

\[
Y_{PM_{10},t|\Phi_{t-1}} \sim N(\mu_{PM_{10},t|\Phi_{t-1}}, \sigma^2_{\varepsilon_{PM_{10},t|\Phi_{t-1}}}),
\]

\[
\mu_{PM_{10},t|\Phi_{t-1}} = \beta_{(1)3} y_{PM_{10},t-1} + \beta_{(1)4} \varepsilon_{PM_{10},t-1} + \beta_{(1)5} \varepsilon_{PM_{10},t-2},
\]

\[
\log \sigma_{\varepsilon_{PM_{10},t|\Phi_{t-1}}} = \beta_{(2)1} + \beta_{(2)2} y_{O_3,t-1} + \beta_{(2)3} y_{O_3,t-2} + \beta_{(2)4} y_{temp,t-1} + \beta_{(2)5} y_{temp,t-2} + \beta_{(2)6} y_{temp,t-3} + \beta_{(2)7} y_{HR,t-1} + \beta_{(2)8} y_{HR,t-2},
\]

where \( y_{PM_{10},t}, y_{O_3,t} \) denote the PM₁₀ and O₃ readings at time \( t \). The conditional mean model is adjusted by humidity and temperature based on a well-known relationship. The correlations between covariates involved with \( \sigma_{\varepsilon_{PM_{10},t|\Phi_{t-1}}} \) are as follows. The tag \texttt{lag} is added to the variable names accordingly.
To estimate (6.2.1) we make use of `Armaxff()` constraining the effect of O₃, temperature and humidity over $\sigma_{PM_{10}}^{t} | \Phi_{t-1}$, and preserving the ARMA(1,2) structure over $\mu_{PM_{10}}^{t} | \Phi_{t-1}$. Further, the drift term is suppressed since the intercepts in Figure (6.2.2), (a) and (b), appear to be in the vicinity of the origin. This may be changed at a later stage of this analysis. The code is the following. In particularly, note that the required lagged values of PM₁₀, O₃, and other explainatories are manually entered in the `formula`. For this, an alternative dataset, called `ap.mx2`, is generated using the function `VGAMextra::WN.lags()`, while the fitted model is saved as `fit.PM10`.

```r
> round(core(tsdata[, c("HR_lag1", "HR_lag2", "temp_lag1", "temp_lag2", "temp_lag3", "O3_lag1", "O3_lag2"))[, 2])

HR_lag1  HR_lag2  temp_lag1  temp_lag2  temp_lag3  O3_lag1  O3_lag2
HR_lag1  1.00    0.86    -0.16    -0.07    0.00   -0.26   -0.21
HR_lag2  0.86    1.00    -0.09    -0.10   -0.02  -0.24   -0.23
temp_lag1 -0.16   -0.09    1.00    0.88    0.77   0.38   0.39
temp_lag2 -0.07   -0.10    0.88    1.00    0.89   0.27   0.39
temp_lag3 0.00   -0.02    0.77    0.89    1.00   0.24   0.30
O3_lag1  -0.26  -0.24    0.38    0.27    0.24   1.00   0.64
O3_lag2  -0.21  -0.23    0.39    0.39    0.30   0.64   1.00
```
## Set up 'ap.mx2' storing lagged values.

```r
n.lag <- 3
ap.mx2 <- transform(ap.mx, 
  PM10_lags = WN.lags(ap.mx[, c("PM10")], n.lag),
  O3_lags = WN.lags(ap.mx[, c("O3")], n.lag),
  temp_lags = WN.lags(ap.mx[, c("temp")], n.lag),
  HR_lags = WN.lags(ap.mx[, c("HR")], n.lag))
colnames(ap.mx2) <- c(colnames(ap.mx), 
  paste("PM10_lag", 1:n.lag, sep = ""),
  paste("O3_lag", 1:n.lag, sep = ""),
  paste("temp_lag", 1:n.lag, sep = ""),
  paste("HR_lag", 1:n.lag, sep = ""))
```

## ARMA(1, 2) with constraints over the forecast variance.

```r
fit.PM10 <- vglm(PM10 ~ O3_lag1 + O3_lag2 + temp_lag1 + temp_lag2 +
  temp_lag3 + HR_lag1 + HR_lag2, 
  ARMAXff(order = c(1, 2), var.arg = FALSE, nodrift = TRUE, 
  type.EIM = "exact", noChecks = TRUE, 
  zero = c("ARcoeff", "MAcoeff")),
  data = ap.mx2)
```

The call `zero = c("ARcoeff", "MAcoeff")` causes `O3_lag1`, `O3_lag2`, `temp_lag1`, `temp_lag2`, `temp_lag3`, `HR_lag1`, and `HR_lag2` to be embodied in the forecast variance model only, while the drift is disregarded with `nodrift = TRUE`. Table 6.2.2 presents a short summary of `fit.PM10` including the estimates and SE's retrieved with `summary()`. More details are given in Appendix A.1. Note that the standard errors from Table 6.2.2 conform with (1.3.15), whereas the VGLTSMs standard errors are computed upon (2.3.14), and are also shown in Appendix A.1.

The function `gamlss::gamlss()` is an alternative for estimating (6.2.1), similarly allowing constraints over $\sigma_{PM10|\Omega_{t-1}}$ but unable to handle moving average terms in the mean model $\mu_{PM10|\Omega_{t-1}}$. Consequently, the lagged errors specifying the MA(2) component must be addressed by other means and then passed into the `gamlss()` call. For the sake of comparison, `gamlss()` is utilised to fit (6.2.1) with errors estimated from two sources: (i) `arima()` and (ii) `ARMAXff()`. Both error sets are plotted in Figure 6.2.3.

The following is the code proposed for this, where `error_lag1` and `error_lag2` denote the aforementioned lagged errors, arranged with `VGAMextra::WN.lags`.

```r
library("gamlss")
## * Errors estimated with arima().
error.arima <- residuals(arima(ap.mx[, "PM10"], order = c(1, 0, 2),
    include.mean = FALSE))
```
Table 6.2.2. Summary of model (6.2.1) fitted with ARMAXff() from VGAMextra.

| Conditional Mean | Estimate | Std. Error | z-value | Pr(>|z|) | Signif. |
|------------------|----------|------------|---------|---------|---------|
| AR coeff ($\beta_{13}$) | 0.99 | 0.008 | 124.243 | 0 | *** |
| MA coeff 1 ($\beta_{14}$) | -0.293 | 0.043 | -6.875 | 0 | *** |
| MA coeff 2 ($\beta_{15}$) | -0.312 | 0.041 | -7.687 | 0 | *** |

## Forecast variance

| Estimate | Std. Error | Pr(>|z|) | Signif. |
|----------|------------|---------|---------|
| Intercept ($\beta_{21}$) | 4.377 | 0.257 | 17.021 | 0 | *** |
| O$_3$ lag 1 ($\beta_{22}$) | -0.003 | 0.001 | -2.676 | 0.007 | ** |
| O$_3$ lag 2 ($\beta_{23}$) | 0.002 | 0.001 | -2.107 | 0.035 | * |
| Temp. lag 1 ($\beta_{24}$) | -0.04 | 0.026 | 1.53 | 0.126 | |
| Temp. lag 2 ($\beta_{25}$) | -0.057 | 0.034 | -1.68 | 0.093 | |
| Temp. lag 3 ($\beta_{26}$) | 0.037 | 0.023 | 1.517 | 0.106 | |
| Rel. hum. lag 1 ($\beta_{27}$) | -0.014 | 0.005 | -2.711 | 0.007 | ** |
| Rel. hum. lag 2 ($\beta_{28}$) | -0.003 | 0.005 | -0.516 | 0.600 | |

## * Errors estimated with ARMAX()

```r
# Error terms estimated with ARMAX()
error.armax <- residuals(fit.PM10, type = "response")
error_lags <- cbind(WN.lags(error.arima, lags = 2), WN.lags(error.armax, lags = 2))

# A) Fitting the model with gamlss() and residuals from arima()
ap.mx3 <- transform(ap.mx2, error_lag1 = error_lags[, 1],
                     error_lag2 = error_lags[, 2])
fit.gamlss.arima <-
gamlss(PM10 ~ PM10_lag1 + error_lag1 + error_lag2, # Model for mu(t)
       sigma.formula = ~ O3_lag1 + O3_lag2 + # Model for sigma(t)
              temp_lag1 + temp_lag2 + temp_lag3 +
              HR_lag1 + HR_lag2,
       family = NO(mu.link = "identity", sigma.link = "log"), #'NO' for normal
       data = ap.mx3, trace = FALSE)

# B) Fitting the model with gamlss() and residuals from arima()
ap.mx3 <- transform(ap.mx2, error_lag1 = error_lags[, 3],
                     error_lag2 = error_lags[, 4])
fit.gamlss.armax <-
gamlss(PM10 ~ PM10_lag1 + error_lag1 + error_lag2, # Model for mu(t)
       sigma.formula = ~ O3_lag1 + O3_lag2 + # Model for sigma(t)
              temp_lag1 + temp_lag2 + temp_lag3 +
              HR_lag1 + HR_lag2,
       family = NO(mu.link = "identity", sigma.link = "log"),
       data = ap.mx3, trace = FALSE)
```

The corresponding summaries for `fit.gamlss.arima`, and `fit.gamlss.armax` are shown in Appendix A.2.

Some goodness-of-fit measures from the estimated models are shown in Table 6.2.3. For a fair comparison, recall that such statistical frameworks differ considerably since ARMAXff() imposes a linear model over the forecast variance which is not amenable
Figure 6.2.3. Estimated errors from model (6.2.1), computed with (a) \texttt{arima()}, and (b) \texttt{ARMAXff()}.

Table 6.2.3. AICs and BICs after fitting model (6.2.1) under two approaches.

<table>
<thead>
<tr>
<th>Package</th>
<th>Modelling function</th>
<th>Family function</th>
<th>Residuals from:</th>
<th>AIC</th>
<th>BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGAMextra</td>
<td>\texttt{vglm()}</td>
<td>ARMAXff()</td>
<td></td>
<td>4215.65</td>
<td>4263</td>
</tr>
<tr>
<td>gamlss</td>
<td>\texttt{gamlss()}</td>
<td>NO()</td>
<td>arima()</td>
<td>4218.64</td>
<td>4270.29</td>
</tr>
<tr>
<td>gamlss</td>
<td>\texttt{gamlss()}</td>
<td>NO()</td>
<td>ARMAXff()</td>
<td>4221.05</td>
<td>4272.71</td>
</tr>
</tbody>
</table>

Overall, moderate effects of short–term atmospheric conditions over the PM$_{10}$ forecast variance, predominantly from ozone, are suggested. Although further analyses including, e.g., cumulative effects might be required, results obtained with \texttt{ARMAXff()} show the relevance of modelling volatility beyond the limited ARCH/GARCH class and variants. This section specially highlights the possibly natural dependence of PM$_{10}$—dispersion on atmospheric variables which may improve the model forecast ability. In particular, we demonstrate VGLTSMs handling constraints on the parameters, a potential aid for non–stationary conditions (non–constant variance) as that suggested in Figure 6.2.1.
6.3 Time series of counts

Time series of counts are accommodated by the VGLTSM framework via the class VGLM–INGARCH (viz. (2.2.16)). This section aims to show and compare the performance of `VGLM.INGARCH()` towards (i) intervention analysis, and (ii) analysis of cumulative effects of air pollution on hospital admissions due to respiratory causes in Hong Kong.

6.3.1 *Campylobacter* infections in Canada: Intervention analysis

We start by analyzing the number of *campylobacterosis* infections in Northern Québec, Canada, reported every 28 days between January 1990 and October 2010. Compared to this, results from `tscount`, `glarma`, and `gamlss` are presented. The series is shown in Figure 6.3.1 retrieved from `campy`, a companion dataset of `tscount`. In particular, Fokianos and Fried (2010), Fokianos and Fried (2012), Liboschik et al. (2016), and Liboschik et al. (2017) present a compendium of intervention analysis techniques using this data set, but restricted to INGARCH models with single intervention effects.

An initial approach may relate to predict the average change rate of *campylobacter* infections over time (denoted here as $\lambda_t$) upon yearly–seasonal effects plus short–term distributed impacts from the last reported period, i.e., serial dependence with no intervention effects. Distributional assumptions such as Poisson and negative binomial seem reasonable, where linear predictor would have the following form (the size parameter is additionally estimated):

$$\log \lambda_t|\phi_{t-1} = \omega + \theta_1 y_{t-1} + \phi_1 \lambda_{t-13}\phi_{t-14}. \quad (6.3.1)$$

Table 6.3.1 shows the AICs, BICs and modelling functions involved after fitting (6.3.1) to `campy` using `VGAMextra`, `tscount`, `glarma`, and `gamlss`. Figure 6.3.2 presents the PIT histograms obtained with `VGAMextra::PIT()`, and Figure 6.3.3 shows the fitted values. Note that the PIT plots, AICs, and BICs indicate that `VGLM.INGARCH()` with the negative binomial assumption produces a superior model–fit compared to the Poisson.

The differences among the AICs/BICs may be explained by modelling discrepancies from the statistical frameworks considered here, specially the log–likelihood optimization. `tsglm()` implements conditional MLE or quasi–MLE using `optim()`, primarily relying on gradients to shape the surface to be optimized through the option `optim.method = "BFGS"`. Further, although its likelihood can be maximized by Fisher scoring or Newton–Raphson, GLARMAs are observation–driven models with
6.3. Time series of counts

Figure 6.3.1. Number of campylobacteriosis infections in Northern Québec reported every 28 days between January 1990 and October 2010.

different modelling features compared to VGLTSMs. The GLARMA linear predictor (state variable) is \( W_t = g(\mathbb{E}(Y_t|\Phi_{t-1})) = g(\lambda_t|\Phi_{t-1}) = \beta^T x_t + O_t + Z_t, \) which depends on an additional offset term, \( O_t, \) while \( Z_t \) is defined by means of residuals \( \xi_t. \) On the other hand, \texttt{gamlss()} implements maximum penalized likelihood estimation using a backfitting–type algorithm.

A more realistic approach would involve \textit{intervention analysis}, as pointed by Fokianos and Fried (2010), Fokianos and Fried (2012), and Liboschik et al. (2016), who identified intervention–influence at times \( t_{84} \) and \( t_{100}. \) The authors utilized the INGARCH–class to explore the campylobacter infections series including single intervention terms. Compared to this, we illustrate the impact of joint–intervention effects on the series by fitting a similar INGARCH model with the addition of interaction terms. Specifically, two (probably influential) readings at the bottom of the series have been selected, \( t_{113}, \) and \( t_{133}, \) occurred around years 1998–2000 (see Figure 6.3.1).

From the pattern shown in Figure 6.3.3, the series may be affected for both ‘shocks’ with possibly singular effects at \( t_{133}, \) hence \( \delta_4 = 0. \) However, the shock \( t_{113} \) lies further in the vicinity of \( t_{100} \) and reasonably around \( t_{84}, \) perhaps influencing the series collectively at exponential factors, where we set a conservative \( \delta_3 = 0.5. \) In addition, we slightly relax the permanent effect at \( t_{84}, \) identified by Fokianos and Fried (2010); Liboschik et al. (2017), while the singular effect at \( t_{100} \) remains, resulting in \( \delta = (0.99, 0, 0.5, 0)^T, \) \( \tau = (84, 100, 113, 133)^T, \) hence the following linear predictor:

\[
\log \lambda_{t|\Phi_{t-1}} = \omega + \delta_1 y_{t-1} + \phi_1 \lambda_{t-13|\Phi_{t-14}} + \sum_{h=1}^{4} \omega_h \cdot \delta_{t-\tau_h} 1(t \geq \tau_h) + \omega_5 \cdot \delta_{t-\tau_1} 1(t \geq \tau_1) \cdot \delta_{t-\tau_3} 1(t \geq \tau_3). \tag{6.3.2}
\]
Table 6.3.1. AICs, BICs after fitting model (6.3.1) and modelling functions utilised.

<table>
<thead>
<tr>
<th>Dist. assumption</th>
<th>Package</th>
<th>Modelling function</th>
<th>Family function</th>
<th>AIC†</th>
<th>BIC†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson</td>
<td>VGAMextra</td>
<td>vglm()</td>
<td>VGLM.INGARCHff()</td>
<td>879.19</td>
<td>888.02</td>
</tr>
<tr>
<td></td>
<td>tcount</td>
<td>tsglm()</td>
<td></td>
<td>880.09</td>
<td>888.92</td>
</tr>
<tr>
<td></td>
<td>glarma</td>
<td>glarma()</td>
<td></td>
<td>913.38</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>gamlss</td>
<td>gamlss()</td>
<td>PO()</td>
<td>881.83</td>
<td>890.65</td>
</tr>
<tr>
<td>Neg. binomial</td>
<td>VGAMextra</td>
<td>vglm()</td>
<td>VGLM.INGARCHff()</td>
<td>817.32</td>
<td>829.09</td>
</tr>
<tr>
<td></td>
<td>tcount</td>
<td>tsglm()</td>
<td></td>
<td>821.37</td>
<td>833.14</td>
</tr>
<tr>
<td></td>
<td>glarma</td>
<td>glarma()</td>
<td></td>
<td>838.15</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>gamlss</td>
<td>gamlss()</td>
<td>NBI()</td>
<td>819.26</td>
<td>831.02</td>
</tr>
</tbody>
</table>

† When available, AICs and BICs computed upon generic funtions AIC() and BIC() from the corresponding packages.

Figure 6.3.2. PIT histograms built upon model (6.3.1), computed with PIT() from VGAMextra, assuming (a) Poisson and (b) negative binomial distributions.

This statistical framework is a special case of the VGLM–INGARCH class supported by VGLM.INGARCHff() (recall that only exponential effects, $\delta_i \in (0, 1)$, are handled currently). The modelling function tcount::tsglm() also accommodates model (6.3.2) except by the interaction term. Both models plus a non–interaction model using VGLM.INGARCHff() were fitted to the campylobacter infections data preserving the negative binomial assumption. The plot of fitted values are shown in Figure 6.3.4; a short summary (only VGLM.INGARCHff()) is given in Table 6.3.3, whilst the AICs/BICs from both modelling framewoks are shown in Table 6.3.2. The summaries of the fitted models are shown in Appendix A.3.

Models from the first and second rows in Table 6.3.2 are comparable since no interaction terms are included. Both outcomes support the conclusions in Liboschik et al.
Figure 6.3.3. Campylobacteriosis infections: Fitted values from model (6.3.1) assuming (a) Poisson and (b) negative binomial response. No intervention effects.

(2017) towards interventions $t_{84}$ and $t_{100}$, but also give marginal to strong evidence of sudden changes, such as special events, influencing the series at $t_{113}$ and $t_{133}$ (see Appendix A.3). However, the series may also see joint–gradual influence from detected interventions examined by the third model, VGLM.INGARCHff() with interaction, showing relevant impacts on the series (possibly unknown interventions) from the proposed $t_{113}$ and $t_{133}$, plus a joint–influence over the series ($\omega_5$), as shown in Table 6.3.3. Finally, note that the interaction version (VGLM–INGARCH) seemingly conveys a finer modelling framework than the model with single intervention effects (Liboschik et al., 2016, 2017).

Nevertheless, the class VGLM–INGARCH is, at present, restricted to modelling and estimation, and still bears a number of demerits. For instance, it lacks methodology for formally detecting interventions. tscount has a superior performance in this line. Also, no forecasting methods upon VGLM–INGARCH have been implemented. However, these features are planned to be incorporated over time, and especially the former is shortly discussed in Section 8.2.3.
Figure 6.3.4. Fitted values from the intervention effects model (6.3.2).

Table 6.3.2. AICs and BICs after fitting (6.3.2) with tsglm() and vglm().

<table>
<thead>
<tr>
<th>Modelling function</th>
<th>Family function</th>
<th>Interaction</th>
<th>AIC</th>
<th>BIC</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>tsglm()</td>
<td>VGLM.INGARCHff()</td>
<td>No</td>
<td>772.83</td>
<td>796.36</td>
<td></td>
</tr>
<tr>
<td>vglm()</td>
<td>VGLM.INGARCHff()</td>
<td>Yes</td>
<td>766.41</td>
<td>792.88</td>
<td>Model introduced with this work, eq. (6.3.2).</td>
</tr>
</tbody>
</table>

Table 6.3.3. Summary of model (6.3.2) fitted with vglm() using VGLM.INGARCHff().

| Coefficient               | Estimate | Std. Error | z–value | Pr(>|z|) | Signif. |
|---------------------------|----------|------------|---------|----------|---------|
| Intercept ($\omega$)      | 1.776    | 0.063      | 28.403  | 0        | ***     |
| Index parameter ($k$)     | 61.95    | 45.565     | 1.36    | 0.174    |         |
| ARcoeff ($\vartheta_1$)   | 0.023    | 0.004      | 6.408   | 0        | ***     |
| MAcoeff1 ($\phi_1$)       | 0.018    | 0.005      | 3.601   | 0        | ***     |
| Interv. effect size 1 ($\omega_1$) | 0.429 | 0.088          | 4.898   | 0        | ***     |
| Interv. effect size 2 ($\omega_2$) | 1.22   | 0.191        | 6.375   | 0        | ***     |
| Interv. effect size 3 ($\omega_3$) | −107.175 | 40.021     | −2.678  | 0.007    | **      |
| Interv. effect size 4 ($\omega_4$) | −0.982 | 0.466          | −2.106  | 0.035    | *       |
| Interaction effect size ($\omega_5$) | 144.237 | 53.634       | 2.689   | 0.007    | **      |
6.3.2 Cumulative effects of air pollution on hospital admissions in Hong Kong

Now VGLM–INGARCH is used for examining the cumulative effects of air pollution on hospital admissions due to respiratory diseases in Hong–Kong. The preceding modelling framework is a GAM–variant called extended–GAM model with cumulative effects, proposed by Xia and Tong (2006b). The dataset analysed by the authors and concerning this section was retrieved from Xia and Tong (2006a) with the following information collected between 1 January 1994 and 31 December 1997 ($n = 1090$):

- Daily emergency hospital admissions for circulatory and respiratory diseases collected from Hospital Information System, from Hong–Kong (we concentrate only on respiratory cases).
- Daily 24–hour average concentrations of NO$_2$ (ppb), O$_3$ (ppb), PM$_{10}$ (ppb), and SO$_2$ (ppb).
- Daily 24–hour mean average of temperature ($^\circ$C), and percentage humidity.
- Weekdays–weekend indicator variables.

This dataset is available in VGAMextra as HKdata, and in this section we particularly concentrate on the series of daily respiratory admissions whose time–plot is shown in Figure 6.3.5. Similar plots of air pollutants and atmospheric variables (temperature and humidity) are presented in Figure 6.3.6.

Estimated by backfitting, the extended–GAM proposed by Xia and Tong (2006b) examines cumulative effects of such air pollutants over hospital admissions due to respiratory diseases (denoted $Y_t$) by incorporating long–term scaled averages of past pollutant levels (up to order $L$) with specific weights determined by smooth functions $g_i$. This GAM–variant has the following form. Note that it also comprises day–of–the–week dummy explanatories, denoted $D_{k,t}$, directing conventional activities with potential for altering daily hospital admissions, e.g., travel behaviour during the weekends:

\[
\begin{align*}
\mathbb{E}(\log Y_t) &= \sum_{k=1}^{6} \beta_k D_{k,t} + g_1 \left( \sum_{\tau=0}^{L} \theta_{1,\tau} N_{t-\tau} \right) + g_2 \left( \sum_{\tau=0}^{L} \theta_{2,\tau} S_{t-\tau} \right) + g_3 \left( \sum_{\tau=0}^{L} \theta_{3,\tau} O_{t-\tau} \right) + g_4 \left( \sum_{\tau=0}^{L} \theta_{4,\tau} P_{t-\tau} \right) + g_5 \left( \sum_{\tau=0}^{L} \theta_{5,\tau} T_{t-\tau} \right) + g_6 \left( \sum_{\tau=0}^{L} \theta_{6,\tau} H_{t-\tau} \right) \\
&= (6.3.3)
\end{align*}
\]

Here, $g_i(\cdot), i = 1, \ldots, 6$ are unknown weighting functions, with $g_1, g_2, g_3$ and $g_4$ monotonic, $\sum_{\tau=0}^{L} \theta_{k,\tau} = 1$ and $0 \leq \theta_{k,0} \leq \theta_{k,1} \leq \cdots \leq \theta_{k,L}$. Here, $N_t$, $S_t$, $O_t$, $P_t$, $T_t$, and $H_t$ denote the daily averages of NO$_2$, SO$_2$, O$_3$, PM$_{10}$, temperature and humidity, respectively, at time $t$. 
Xia and Tong (2006b) explored different case scenarios of (6.3.3), especially for large $L$ (e.g. $L \geq 300$). However, the central concern is that it relies on an excessive number of parameters this model has, most likely leading to several modelling issues. While clearly the extended-GAM accounts for serial dependence, the inclusion of too many predictors may be ignoring significant intercorrelation structures and possibly hiding covariates with good predictive capabilities. In addition, customary modelling practices such as model interpretation or model selection, may draw poor conclusions when too many predictors are involved, for instance, the modelling–quality criterion AICs/BICs, which indeed penalize by the number of parameters. Furthermore, special care may be required with small datasets, which might reflect only peculiarities and variability of the sample, rather than population features.

Instead, \texttt{VGLM.INGARCHff()} represents a convenient choice to mimic the heavily parametrized extended–GAM. Specified by daily counts, the series of hospital admissions very likely conform with either a Poisson or a negative binomial process with specific linear structures on the mean–response, $\lambda_{t|\Phi_{t-1}}$. In fact, the cumulative effects of outdoor exposure to air pollutants, as well as stochastic seasonality can be simultaneously investigated with \texttt{VGLM.INGARCHff()} by regressing on lagged conditional means, $\lambda_{t-j_k|\Phi_{t-j_k-1}}$, for arbitrary but convenient lags $t - j_k$ (rather than extensively lagged components). We specially focus on the fixed $j_k = (7, 15, 30, 90, 180, 366)^T$ to explore weekly, fortnightly, 1–month, 3–months, 6–months, and yearly cumulative effects. And to somewhat account for serial autocorrelation we choose an ARMA(1, 1) structure over $\lambda_{t|\Phi_{t-j_k}}$, with the addition of “weekends” indicator variables, peaking, e.g., travel behaviour, whilst the short–term influence of pollutants and atmospheric conditions on hospital admissions is explored, say, through past 2–day exposures. With
these assumptions, the following VGLM–INGARCH variant is proposed to statistically depict the hospital admissions for respiratory diseases in Hong Kong:

\[
Y_t | \Phi_{t-1} \sim \mathcal{F}(\lambda_t | \phi_{t-1}, \cdots; \Phi_{t-1}),
\]

\[
\log \lambda_t | \phi_{t-1} = \omega + \delta_1 D_{5,t} + \delta_2 D_{6,t} + \theta_1 Y_{t-1} + \phi_1 \lambda_{t-1} | \phi_{t-1} + \\
\sum_{s=1}^{6} \phi_s + 1 \lambda_{t-j,s} | \phi_{t-j,s-1} + \sum_{j=0}^{2} \beta_1,j O_{3,t-j} + \sum_{k=0}^{2} \beta_2,k PM_{10,t-k} + \\
\sum_{j=1}^{2} \beta_3,j NO_{2,t-j} + \sum_{k=1}^{2} \beta_3,k SO_{2,t-k} + \sum_{l=1}^{2} \beta_4,l T_{t-l} + \sum_{\tau=1}^{2} \beta_4,\tau H_{t-\tau}.
\]

This model was fitted to the data adopting a Poisson and negative binomial response (\(\mathcal{F}\)) using \texttt{VGLM.INGARCH()} and the following code. The data frame \texttt{HKdata.2} comprises the required lagged values, that have been arranged with \texttt{WN.lags()}. 

Figure 6.3.6. Series of pollutants and atmospheric variables, Hong Kong (Jan 1995–Dec 1997): (a) Ozone (ppb), (b) PM\(_{10}\) (ppb), (c) NO\(_2\) (ppb), (d) SO\(_2\) (ppb), (e) Temperature (°C), (f) Humidity (%).
Table 6.3.4. Estimated percentage change in daily Hong Kong hospital admissions from model (6.3.4). Response assumed as NB.

| Effects                        | Parameter | % change† | Std. error | Pr(>|z|) | Signif |
|--------------------------------|-----------|-----------|------------|---------|--------|
| Ozone (1-day lagged)           | $\beta_{1,0}$ | 0.914%    | 2.2×10^{-4} | 0       | ***    |
| PM$_{10}$ (1-day lagged)       | $\beta_{2,0}$ | -0.588%   | 2.8×10^{-4} | 0.034   | *      |
| Cumulative 1-day lagged        | $\phi_1$  | 1.167%    | 2.4×10^{-4} | 0       | ***    |
| Cumulative 7-day lagged        | $\phi_2$  | 1.298%    | 1.8×10^{-4} | 0       | ***    |
| Cumulative 15-day lagged       | $\phi_3$  | 0.08%     | 1.5×10^{-4} | 0.604   | **     |
| Cumulative 1-month             | $\phi_4$  | 0.04%     | 1.1×10^{-4} | 0.697   | .      |
| Cumulative 3-months            | $\phi_5$  | 0.13%     | 9×10^{-5}   | 0.14    | .      |
| Cumulative 6-months            | $\phi_6$  | -0.12%    | 6×10^{-5}   | 0.05    | *      |
| Cumulative 12-months           | $\phi_7$  | 0.11%     | 6×10^{-5}   | 0.045   | **     |

† Per 10-units increase, assuming negative binomial response.

n.lags <- 3
HKdata.2 <- transform(HKdata,
  lag.no2 = WN.lags(HKdata[, "no2"], lags = n.lags),
  lag.so2 = WN.lags(HKdata[, "so2"], lags = n.lags),
  lag.rsp = WN.lags(HKdata[, "rsp"], lags = n.lags),
  lag.o3 = WN.lags(HKdata[, "o3"], lags = n.lags),
  lag.temp = WN.lags(HKdata[, "temp"], lags = n.lags),
  lag.hum = WN.lags(HKdata[, "hum"], lags = n.lags))

names(HKdata.2) <- c(colnames(HKdata),
  paste("no2lag", 1:n.lags, sep=""),
  paste("so2lag", 1:n.lags, sep=""),
  paste("rsplag", 1:n.lags, sep=""),
  paste("o3lag", 1:n.lags, sep=""),
  paste("templag", 1:n.lags, sep=""),
  paste("humlag", 1:n.lags, sep=""))

fit.HK <- vglm(resp ~ fri + sat + o3 + rsp +
  no2lag1 + so2lag1 + rsplag1 + o3lag1 + templag1 + humlag1 +
  no2lag2 + so2lag2 + rsplag2 + o3lag2 + templag2 + humlag2,
  family = VGLM.INGARCHff(dist.type = "negbinomial",
    Order = c(1, 1), link = "loglink",
    interventions = list(),
    lagged.fixed.means = c(7, 15, 30, 90, 180, 366)),
  data = HKdata.2)

The resulting PIT histograms are shown in Figure 6.3.8 while the original data compared to the fitted values from (6.3.4), for response as Poisson and NB, are shown in Figure 6.3.7. Based on the PITs, the data seems to best conform with the NB assumption, which is the approach underlying the short summary presented in Table 6.3.4.

Xia and Tong (2006b) fitted model (6.3.3) for $L = 365$ and found that O$_3$ and the meteorological predictors were the main contributors towards hospital admissions due
6.3. Time series of counts

Figure 6.3.7. Fitted values of Hong Kong hospital admissions due to respiratory diseases from model (6.3.4) assuming the response is (a) NB and (b) Poisson.

Figure 6.3.8. PIT histograms build on the estimated model (6.3.4), computed with VGAMextra::PIT(); (a) Poisson, (b) NB.

to respiratory problems in Hong–Kong. PM$_{10}$, was found to be confounded with age, with worse effects for elderly people. However, the distributional assumption on the response is vaguely specified. The plot of fitted values retrieved from Xia and Tong (2006b) is shown in Appendix A.4. The results from VGLM.INGARCHff(), also shown in Appendix (A.4) saved as fit22_nb, report similar short–term effects of O$_3$ on the same outcome (i.e., daily respiratory admissions), as well as short term impact of changes in temperature and humidity (1–day or 2–day lags). An ambiguous short–term effect of PM$_{10}$ is also reported, which may be confounded with other covariates, such as age–group or socio–economic status. In addition, Xia and Tong (2006b) highlighted the ‘weekend effect’ as having less impact than ‘weekdays’, which was also confirmed by VGLM.INGARCHff().
6.4 RR–VGLMs towards systems of CTS

Systems of cointegrated time series via order \((u, v)\)–ECMs and the Engle–Granger procedure have shown to be amenable to VGLTSMs, however, as discussed in Chapter 5, the number of coefficients, \(L\), to be estimated from ECMs steadily grows as the order \((u, v)\) increases. In this section we show how a variant of VGLMs may remedy situations in CTS analysis where \(L\) is large, namely the sub–class of RR–VGLMs (Section 1.3.2). This approach represents an alternative to that described in Chapter 5 for higher orders \(u, v\), yet preserving the core information relevant to purposes such as forecasting from CTS.

Similar to VGLMs estimated with VGAM::vglm(), RR–VGLMs are handled by the modelling function VGAM::rrvglm(), and also are able to accommodate VGLTSMffs, e.g., ECM.EngleGran(), as a result of RR–VGLMs being a sub–class of VGLMs.

Concerning time series analysis with RR–VGLMs, recall that the VGLM–linear predictor, given by

\[
\eta_t = \mathbf{B}^T \mathbf{x}_t,
\]

(6.4.1)

with \(\mathbf{B}\) an \(L = M \times p\) matrix of coefficients, and a \(p\)–dimensional \(\mathbf{x}_t\) covariates, is partitioned into two components, as in (1.3.7), acquiring the form:

\[
\eta_t = \mathbf{B}_1^T \mathbf{x}_{1,t} + \mathbf{A} \mathbf{C}^T \mathbf{x}_{2,t},
\]

(6.4.2)

where \(\mathbf{x}_t = (\mathbf{x}_{1,t}, \mathbf{x}_{2,t})^T, \dim(\mathbf{x}_t) = p_i, \sum p_i = p, i = 1, 2\); and \(\mathbf{A}^T, \mathbf{C}\) are thin matrices, both of rank \(R \ll \min(M, p_2)\), with \(\mathbf{B}_2^T = \mathbf{A} \mathbf{C}^T\). Here, the reduced rank regression is directly applied to \(\mathbf{B}_2\).

To illustrate the benefits of RR–VGLMs in this context, consider those two cointegrated random walks addressed in Section 5.3.3, \(\mathbf{y}_t = (\mathbf{x}_t, \mathbf{y}_t)^T\), generated as follows:

\[
\begin{align*}
\mathbf{x}_t &= \mathbf{x}_{t-1} + \mathbf{\epsilon}_{t,1}, \\
\mathbf{y}_t &= \beta_0 + \beta_1 \mathbf{x}_t + \mathbf{\epsilon}_{t,2},
\end{align*}
\]

(6.4.3)

where \(\mathbf{\epsilon}_t = (\mathbf{\epsilon}_{t,1}, \mathbf{\epsilon}_{t,2})^T \sim \mathcal{N}_2(0, \Sigma), \Sigma = \begin{pmatrix}
\sigma_{\mathbf{\epsilon}_{t,1}}^2 & \sigma_{\mathbf{\epsilon}_{t,1}\mathbf{\epsilon}_{t,2}} \\
\sigma_{\mathbf{\epsilon}_{t,1}\mathbf{\epsilon}_{t,2}} & \sigma_{\mathbf{\epsilon}_{t,2}}^2
\end{pmatrix}\), with \(\sigma_{\mathbf{\epsilon}_{t,1}} = \exp(2 \cdot \log(1.5)), \sigma_{\mathbf{\epsilon}_{t,2}} = \exp(2 \cdot \log(0.75)), \rho = 0.75, \) and \((\beta_0, \beta_1)^T = (0.0, 0.60)^T\). In Chapter 5, the dynamic behaviour of \(\mathbf{y}_t\) is addressed upon the following VGLM–ECM, for \(u = v = 1\), which conforms with an ECM(1, 1) in Engle and Granger (1987):
\[
(\nabla x_t, \nabla y_t)^T \sim N_2((\mu_{\nabla x_t}, \mu_{\nabla y_t})^T, \Sigma),
\]
\[
\mu_{\nabla x_t} = \phi_0 + \gamma_1 \hat{z}_{t-1} + \sum_{i=1}^u \phi_{i,1} \nabla y_{t-i} + \sum_{j=1}^v \phi_{j,2} \nabla x_{t-j},
\]
\[
\mu_{\nabla y_t} = \psi_0 + \gamma_2 \hat{z}_{t-1} + \sum_{i=1}^u \psi_{i,1} \nabla x_{t-i} + \sum_{j=1}^v \psi_{j,2} \nabla y_{t-j},
\]
where \( \mu_{\nabla x_t} = \mathbb{E}(\nabla x_t | \Phi_{t-1}) \), and \( \mu_{\nabla y_t} = \mathbb{E}(\nabla y_t | \Phi_{t-1}) \), specify the vector response \((\nabla x_t, \nabla y_t)^T\). The notation \( \nabla x_t | \Phi_{t-1} = \nabla x_t \), \( \nabla y_t | \Phi_{t-1} = \nabla y_t \) is adopted.

Model (6.4.4) involves \( L = 11 \) coefficients to be estimated, and incrementing \( u, v \), to \( u = v = 2 \), will imply \( L = 15 \) coefficients to be estimated, viz. \( \phi_0, \gamma_0, \gamma_k, \phi_{i,k}, \psi_{j,k}, k = 1,2; i,j = 1,2 \), plus three entries of the covariance matrix \( \Sigma \). Utilizing \texttt{ECM.EngleGran()} we readily fit such VGLM–ECM(2,2) to the series \( y_t \) with the following code, which also gives the estimates including the estimated co-integrated vector:

```r
> coint.Data <- data.frame(xt = xt, yt = yt)
> fit1.coint.VGAMextra <- vglm(cbind(xt, yt) ~ 1,
   ECM.EngleGran(ecm.order = c(2, 2),
   resids.pattern = "intercept",
   zero = c("var", "cov"),
   data = coint.Data)
```

The residuals appear to be free of unit roots.

Estimated co-integrating vector:

<table>
<thead>
<tr>
<th>betaY2 (Intercept)</th>
<th>betaY1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00000</td>
<td>-0.53770</td>
</tr>
<tr>
<td></td>
<td>-0.64103</td>
</tr>
</tbody>
</table>

Final sample size: 97

During the fitting process, required lagged values of both response and predictors, are internally computed and allocated to the vector \( x_t \) (cf. (6.4.1)). In this example, \( x_t \) comprises the following information for every observation \( t = 1, \ldots, 97 \):

\[
x_t^T = [1 \mid \text{errors.cointLag1} \mid \text{difixLag1} \mid \text{difyLag1} \mid \text{difixLag2} \mid \text{difyLag2}].
\]

Here \( 1 \) signifies the intercept, \( \text{errors.cointLag1} \) the residuals of the regression \( y_t \sim x_t \), and, \( \text{difixLag1}, \text{difixLag2}, \text{difyLag1}, \) and \( \text{difyLag2} \), carries information pertaining \( \nabla x_{t-i}, i = 1,2 \), and \( \nabla y_{t-j}, j = 1,2 \).
However, RR–VGLMs may simplify computational efforts derived from the above analysis by applying reduced rank regression over \( B_2 \), the matrix carrying the coefficients pertaining to the “covariates” in \( x_{2,t} \). In this example, it has the form:

\[
x_{2,t}^T = [ \text{errors.cointLag1} | \text{difxLag1} | \text{difyLag1} | \text{difxLag2} | \text{difyLag2} ].
\]

The matrix \( B_1 \) in (6.4.2) then carries the intercepts \( \beta_{(j)1}, j = 1, \ldots, 5 \), whereas \( x_{1,t} \) becomes a vector of 1’s (matching the intercepts), thus producing

\[
\eta_t = B_1^T x_{1,t} + B_2^T x_{2,t} = \begin{pmatrix} \beta_{(1)1} \\ \vdots \\ \beta_{(5)1} \end{pmatrix} + AC^T x_{2,t}.
\] (6.4.5)

Therefore we can “summarize” our VGLM–ECM(2, 2) through the matrices \( B_1, A, \) and \( C \) (as \( AC^T = B_2 \)), which are estimated along with the fit of an RR–VGLM to the data using \texttt{rrvglm()}\), as per the code below.

```r
### An ECM(2, 2), where reduced--rank regression is applied. rrvglm() is used.
> rrfit.coint <- rrvglm(cbind(xt, yt) ~ 1, 
  ECM.EngleGran(ecm.order = c(2, 2), 
  resids.pattern = "intercept", 
  zero = NULL),str0 = c(3, 4, 5), 
  data = coint.Data)

The residuals appear to be free of unit roots.
Estimated co-integrating vector:
betaY2 (Intercept) betaY1
1.00000 -0.53770 -0.64103
Final sample size: 97

Finally, we restore and compare the estimated coefficients from \texttt{fit1.coint.VGAMextra} (without RRR) and \texttt{rrfit.coint1} (with RRR):

```
C matrix:

<table>
<thead>
<tr>
<th>latvar</th>
<th>ErrorsLag1</th>
<th>diffy1Lag1</th>
<th>diffy1Lag2</th>
<th>diffy2Lag1</th>
<th>diffy2Lag2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.2007</td>
<td>-0.1269</td>
<td>0.0282</td>
<td>0.1941</td>
<td>-0.0229</td>
</tr>
</tbody>
</table>

B1 matrix:

<table>
<thead>
<tr>
<th>Diff1</th>
<th>Diff2</th>
<th>loglink(var1)</th>
<th>loglink(var2)</th>
<th>cov12</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.115</td>
<td>0.0751</td>
<td>0.879</td>
<td>1.01</td>
</tr>
</tbody>
</table>

### I) Retrieve the estimated coefficients from 'fit1.coint.VGAMextra'.

***No RRR applied here. This is the usual 'B' in eta_t = B^T x.***

```r
t(coef(fit1.coint.VGAMextra, matrix = TRUE))
```

### II) Compute the estimated 'A * C^T' from 'rrfit.coint', where RRR is applied

***to 'B_2', as in eq. (6.4.5), then compare with I).***

```r
(matrix.rr@A %*% t(matrix.rr@C))
```

### III) Compare the intercepts: B_1 hat in (6.4.5) vs. 'fit1.coint.VGAMextra'

```r
t(coef(fit1.coint.VGAMextra, matrix = TRUE)[-1, ])
```

```r
(matrix.rr@B1)
```
Despite mild discrepancies between the estimated coefficients, we have reproduced the CTS analysis spanning in Chapter 5 using the RR–VGLM class with the advantage of handling and estimating fewer coefficients and most importantly, preserving the VGLM structure which conveys the ability of further analysis in areas such as forecasting.

6.5 Summary

This chapter has illustrated VGLTSMs and VGLMs as alternatives to preceding modelling structures for time series analysis. Conveniently, VGLTSMs confer a number of modelling features via covariate–specific linear predictors allowing finer analysis in the autocorrelation structure. We have especially shown parameter constraints over the forecast variance, not handled by other software to our knowledge, in \textit{R}. Real data has been used for these purposes.

Similarly, the class of ECM–like models for cointegration (in the sense of \textit{Engle and Granger (1987)}) has shown to be amenable to RR–VGLMs and advantageous for over–parametrized cases of ECMs.
Chapter 7

Miscellaneous VGLM Topics

7.1 Introduction

Along with the work from the previous chapters, additional methodology based on VGLMs has been developed in a few other directions, as follows:

(I) On estimating VGLMs with two-parameter links, and

(II) (a) On mean modelling and (b) quantile modelling of several 1-parameter distributions.

This chapter gives details on the work developed towards (I)–(II), as well as some examples using simulated data that are to be compared with other software. Regarding (I), we particularly illustrate the special case of the NB-C_2-2 model and its 2-parameter canonical link. Like the VGLTSM work, this chapter is supported by software in VGAMextra.

7.2 On estimating VGLMs with 2–parameter link functions

To commence, we consider a central topic on VGLM modelling that can be thought of as an extension of the VGLM–framework towards handling link functions with 2 parameters. This is a novel methodology that has been scarcely exploited in special cases (see, e.g., Hilbe, 2011) but is shown to confer advantages to VGLMs & VGAMs, for instance, to estimate the negative binomial NB–C_2–2 variant having the 2–parameter canonical link (see Yee, 2015, p. 325, where k is modelled as intercept–only).

Following from Section 1.3, the VGLM log–likelihood is

\[ \ell(\eta; x) = \sum_{i=1}^{n} w_i \ell_i\{\eta_1(x_i), \ldots, \eta_M(x_i)\}, \]
depending on \( M \) linear predictors \( \eta_j = \beta_j^T x \), that is valid for statistical models of the form (viz. (1.3.1))

\[
F(y|x; B) = h(y, \eta_1, \ldots, \eta_M, \Phi; x),
\]

conditional on explanatories \( x \) with (possibly multivariate) response \( y \).

Typically, the \( \eta_j \)'s are applied to individual parameters \( \theta_j \) of any distribution, as \( g_j(\theta) = \eta_j \), giving place to the corresponding log–likelihood (1.3.8). When \( M = 2 \) the linear predictor acquires the form

\[
\eta_i = \left( \begin{array}{c} \eta_1(x_i) \\ \eta_2(x_i) \end{array} \right) = \left( \begin{array}{c} g_1(\theta_{1i}) \\ g_2(\theta_{2i}) \end{array} \right) = \left( \begin{array}{c} \beta_{1i}^1 \\ \beta_{1i}^2 \end{array} \right) \left( \begin{array}{c} \beta_{2i}^1 \\ \beta_{2i}^2 \end{array} \right) x_i, \quad i = 1, \ldots, n; \quad (7.2.1)
\]

and then the VGLM log–likelihood reduces to

\[
\ell(\eta;x) = (\eta_1(\theta_1), \eta_2(\theta_2); x) = \sum_{i=1}^n w_i \{ \eta_1(x_i), \eta_2(x_i) \},
\]

where the score vector and the \( ij \)th entry of the EIMs (where available) are derived accordingly from

\[
\frac{d\ell}{d\eta} = \left( \begin{array}{c} \frac{d\ell}{d\eta_1} \\ \frac{d\ell}{d\eta_2} \end{array} \right) = \left( \begin{array}{c} \frac{\partial \ell}{\partial \eta_1} \\ \frac{\partial \ell}{\partial \eta_2} \end{array} \right), \quad \text{and} \quad I(\theta)_{j,k} = -\mathbb{E} \left[ \frac{\partial^2 \ell(\eta_1, \eta_2)}{\partial \eta_j \partial \eta_k} \right]. \quad (7.2.2)
\]

Due to its connection to \( \eta_j = g_j(\theta_j) \), we must account for a few chain rule formulae, in addition, to compute (7.2.2), viz.

\[
\frac{\partial \ell}{\partial \eta_j} = \frac{\partial \ell}{\partial \theta_j} \cdot \frac{\partial \theta_j}{\partial \eta_j}, \quad (7.2.3)
\]

\[
-\mathbb{E} \left[ \frac{\partial^2 \ell}{\partial \eta_j^2} \right] = -\mathbb{E} \left[ \frac{\partial^2 \ell}{\partial \theta_j^2} \right] \left( \frac{\partial \theta_j}{\partial \eta_j} \right)^2,
\]

\[
-\mathbb{E} \left[ \frac{\partial^2 \ell}{\partial \eta_j \eta_k} \right] = -\mathbb{E} \left[ \frac{\partial^2 \ell}{\partial \theta_j \partial \theta_k} \right] \left( \frac{\partial \theta_j}{\partial \eta_j} \right) \left( \frac{\partial \theta_k}{\partial \eta_k} \right), \quad j, k = 1, \ldots, M.
\]

However, this methodology restricts each linear predictor to model a single parameter, thus impeding the development of meaningful theoretical results for VGLMs/VGAMs. The NB-C_{2-2} variant with parameters \( \theta_1 = \mu \) and \( \theta_2 = k \), i.e., the mean and size parameters to be estimated, with canonical link

\[
\log \frac{\theta_1}{\theta_1 + \theta_2},
\]

represents a prominent example here. From VGAM 1.0-6 onwards, the family function
to estimate NB-C2-2 is `negbinomial()` where the current situation involves two linear predictors, given by

\[ \eta_1(\theta_1, \theta_2; \mathbf{x}) = \beta_1^T \mathbf{x} = \log \frac{\theta_1}{\theta_1 + \theta_2} = g_1(\theta_1, \theta_2; \mathbf{x}), \]  
\[ \eta_2(\theta_2; \mathbf{x}) = \beta_2^T \mathbf{x} = \log \theta_2 = g_2(\theta_2; \mathbf{x}). \]  

(7.2.4)

Our primary interest here is to investigate how the VGLM–loglikelihood (1.3.8) ought to be modified to accommodate VGLMs such as in (7.2.4), or even more generally if the linear predictors involved \( M > 2 \) parameters. At present, we have implemented the theory and methodology for \( M = 2 \) in the next section. Prior to this work, the NB canonical link had been unsatisfactorily implemented (see, e.g., Section 11.3.4 in Yee, 2015, which concerns VGAM 1.0-0 and earlier).

### 7.2.1 The modified VGLM–framework

The effect of (7.2.4) on the VGLM/VGAM–framework is reflected in the score vector and the working weight matrices, the essential requirements for IRLS–Fisher scoring. Consider the parametric–conditional statistical model \( f(y; \Theta, \mathbf{x}), \Theta = \{\theta_1, \theta_2\} \subseteq \mathbb{R}^2 \), with response \( y = \{y_1, \ldots, y_n\} \) and covariates \( \mathbf{x} \). Let us assume that our inference–interest focusses on smooth transformations of the elements of \( \Theta \), via the 2–parameter linear predictors:

\[ \eta_{G_1}(\mathbf{x}) = G_1(\theta_1, \theta_2), \quad \eta_{G_2}(\mathbf{x}) = G_2(\theta_1, \theta_2), \]  

(7.2.5)

for known smooth functions \( G_k, k = 1, 2 \). At least the first two derivatives of \( G_k \) should be available, as with ordinary link functions, and, in addition, the inverse expressions and the first two derivatives must be available too. This means that the system (7.2.5) should have a unique solution, say \((\theta_{1,\eta_{G_1},\eta_{G_2}}, \theta_{2,\eta_{G_1},\eta_{G_2}})^T\) for every pair \((\eta_{G_1}, \eta_{G_2})\), such that the inverse functions \( \theta_{1,\eta_{G_1},\eta_{G_2}} \) and \( \theta_{2,\eta_{G_1},\eta_{G_2}} \) are feasible. This implies that the associated Jacobian must be of full rank in the entire parameter space. For simplicity, we will use the following notation

\[ \theta = (\theta_{1,\eta_{G_1},\eta_{G_2}}, \theta_{2,\eta_{G_1},\eta_{G_2}})^T = (\theta_{G_1}^1, \theta_{G_2}^2)^T \]  

(7.2.6)

to denote the inverse functions, as well as \( \eta_{G_1, G_2} = (\eta_{G_1}, \eta_{G_2})^T \).

Upon these conditions, the log–likelihood characterizing such VGLMs has the following functional form:

\[ \ell = \ell(\eta_{G_1}(\theta_1, \theta_2), \eta_{G_2}(\theta_1, \theta_2); y, \mathbf{x}), \]  

(7.2.7)

and, fundamentally, the only modification to the VGLM–framework to accommodate (7.2.5) is to adapt the expressions for the score vector and the EIMs accordingly.
7.2.1.1 The modified score vector

As \( \ell = \ell(\eta_{G_1}(\theta_1), \eta_{G_2}(\theta_2); y, x) = \ell(\eta_{G_1, G_2}; y, x) \), the expressions for the derivatives \( d\ell/d\eta \) in (7.2.2) are obtained from the chain rule, viz.

\[
\frac{d\ell}{d\eta_{G_1, G_2}} = \left( \frac{d\ell}{d\eta_{G_1}} \right) \cdot \frac{d\ell}{d\theta^T}, \quad (7.2.8)
\]

Ordinarily, by reason of the VGLM–framework dependency on 1–parameter link functions of the form \( \eta_{G_j}(x) = G_j(\theta^k_{G_j}), j = 1, 2, \) one has

\[
\left( \frac{d\theta^T}{d\eta} \right)^T = \begin{pmatrix} \frac{d\theta^1_{G}}{d\eta_{G_1}} & 0 \\
0 & \frac{d\theta^2_{G}}{d\eta_{G_2}} \end{pmatrix} \quad \text{and} \quad \frac{d\ell}{d\theta} = \begin{pmatrix} \frac{\partial\ell}{\partial\theta^1_{G}} \\
\frac{\partial\ell}{\partial\theta^2_{G}} \end{pmatrix}, \quad (7.2.9)
\]

yielding

\[
\frac{d\ell}{d\eta_{G_1}} = \frac{d\ell}{d\theta^1_{G}} \frac{d\theta^1_{G}}{d\eta_{G_1}} = \frac{\partial\ell}{\partial\theta^1_{G}} \frac{d\theta^1_{G}}{d\eta_{G_1}}, \quad \text{and} \quad (7.2.10)
\]

\[
\frac{d\ell}{d\eta_{G_2}} = \frac{d\ell}{d\theta^2_{G}} \frac{d\theta^2_{G}}{d\eta_{G_2}} = \frac{\partial\ell}{\partial\theta^2_{G}} \frac{d\theta^2_{G}}{d\eta_{G_2}},
\]

conforming with (7.2.2). The \( \partial\ell/\partial\theta^k_{G} \), \( k = 1, 2 \), are obtained from the usual log–likelihood \( \ell(\theta_1, \theta_2; y, x) \), while \( \partial\theta^k_{G}/\partial\eta_{G_k} \), should be obtained from the inverse \( G_j^{-1} = \theta^k_{G}(\eta_{G_k}) \), guaranteed by means of the monotonicity of \( G \).

Crucially however, for 2–parameter VGLM–links the derivatives (7.2.9) must account for the 2–parameter dependency of the log–likelihood, as expressed in (7.2.7), giving place to the adjusted components of the score vector (7.2.9), as follows:

\[
\left( \frac{d\theta^T}{d\eta} \right) = \begin{pmatrix} \frac{d\theta^1_{G}}{d\eta_{G_1}} & \frac{d\theta^2_{G}}{d\eta_{G_1}} \\
\frac{d\theta^1_{G}}{d\eta_{G_2}} & \frac{d\theta^2_{G}}{d\eta_{G_2}} \end{pmatrix} \quad \text{and} \quad \frac{d\ell}{d\theta} = \begin{pmatrix} \frac{d\ell}{d\theta^1_{G}} \\
\frac{d\ell}{d\theta^2_{G}} \end{pmatrix}, \quad (7.2.11)
\]

that is,

\[
\frac{d\ell}{d\eta_{G_1}} = \frac{d\ell}{d\theta^1_{G}} \frac{d\theta^1_{G}}{d\eta_{G_1}} + \frac{d\ell}{d\theta^2_{G}} \frac{d\theta^2_{G}}{d\eta_{G_1} \eta_{G_2}}, \quad \text{and} \quad (7.2.12)
\]

Moreover, note that the log–likelihood (7.2.7) involves at least three inference case scenarios in terms of the number of parameters pertaining to the linear predictors, for
7.2. On estimating VGLMs with 2–parameter link functions

Table 7.2.1. Expressions of the score vector for 2–parameter VGLMs considering the three compositions of the linear predictors (7.2.5).

<table>
<thead>
<tr>
<th>Linear predictors, $(\eta_{G_1, G_2})$</th>
<th>Inverse–$(\eta_{G_1, G_2})$ system†</th>
<th>Composite–ℓ</th>
<th>Modified expressions for $d\ell/d\theta^1_G$, $d\ell/d\theta^2_G$ in (7.2.12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta_{G_1} = G_1(\theta_1, \theta_2)$, $\theta = (\theta^1_G, \theta^2_G)^T$</td>
<td>$\ell(\theta^1_G, \theta^2_G; y)$</td>
<td>$d\ell/d\theta^1_G = \frac{\partial \ell}{\partial \theta^1_G}$, $d\ell/d\theta^2_G = \frac{\partial \ell}{\partial \theta^2_G}$</td>
<td></td>
</tr>
<tr>
<td>$\eta_{G_2} = G_2(\theta_2)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\eta_{G_1} = G_1(\theta_1, \theta_2)$, $\theta = (\theta^1_G, \theta^2_G, \theta^3_G)^T$ $\ell(\theta^1_G, \theta^2_G, \theta^3_G; y)$ $d\ell/d\theta^1_G = \frac{\partial \ell}{\partial \theta^1_G}$, $d\ell/d\theta^2_G = \frac{\partial \ell}{\partial \theta^2_G}$, $d\ell/d\theta^3_G = \frac{\partial \ell}{\partial \theta^3_G}$ |
| $\eta_{G_2} = G_2(\theta_1, \theta_2)$ |

$\eta_{G_1} = G_1(\theta_1, \theta_2)$, $\theta = (\theta^1_G, \theta^2_G, \theta^3_G)^T$ $\ell(\theta^1_G, \theta^2_G, \theta^3_G; y)$ $d\ell/d\theta^1_G = \frac{\partial \ell}{\partial \theta^1_G}$, $d\ell/d\theta^2_G = \frac{\partial \ell}{\partial \theta^2_G}$, $d\ell/d\theta^3_G = \frac{\partial \ell}{\partial \theta^3_G}$ |
| $\eta_{G_2} = G_2(\theta_1, \theta_2)$ |

† Assuming that $\eta_{G_1, G_2}$ has a unique solution $(\theta^1_G, \theta^2_G)^T$ for every $(\eta_{G_1}, \eta_{G_2})^T$.

instance, $\eta_{G_1, G_2}(x) = (\eta_{G_1}, \eta_{G_2})^T$ with $\eta_{G_1}(x) = G_1(\theta^1_G)$, and $\eta_{G_2}(x) = G_2(\theta^2_G)$. Under the assumption that this system poses a unique solution $(\theta^1_G, \theta^2_G)^T$ for every $(\eta_{G_1}, \eta_{G_2})^T$, the inverse functions

$$
\theta^1_G = G_1^{-1}(\eta_{G_1}) \quad \text{and} \quad \theta^2_G = G_2^{-1}(\eta_{G_2}) \quad (7.2.13)
$$

are tractable, producing a re–parametrized log–likelihood, as

$$
\ell(\theta^1_G, \theta^2_G; y, x) = \ell(\theta^1_G, \theta^2_G; y, x). \quad (7.2.14)
$$

This form of $\ell$ signifies a composite–VGLM log–likelihood, and represents a central expression in this example (and in this work) as it implies $d\ell/d\theta^j_G$ in (7.2.12) to be derived in terms of the chain rule, while $d\theta^j_G/d\eta_{G_k}$, $j, k = 1, 2$, can be obtained from the inverse–$(\eta_{G_1, G_2})$ in (7.2.13). This and all other possible options for the linear predictor $\eta_{G_1, G_2}$ in this context are summarized in Table 7.2.1, which also gives the modified expressions for $d\ell/d\theta^1_G$, $d\ell/d\theta^2_G$ in (7.2.11) and (7.2.12), compared to the ordinary derivatives (7.2.10).

7.2.1.2 The modified working weight matrices

The modified expressions for the EIMs derive similarly from the chain rule, as an extension of (7.2.2) and (7.2.3). Note that the OIM is the second–order derivative of $\ell(\eta_{G_1, G_2}; y, x)$ given by
the case when parameter dependency of the linear predictors stated in (7.2.5). For instance, consider where the chain rule also yields

\[
\frac{d^2 \ell}{d \eta_1^2} = \frac{d^2 \ell}{d \eta_1^2} + \left\{ \sum_j \frac{\partial \ell}{\partial \theta_j} \cdot \frac{\partial^2 \theta_j}{\partial \eta_1 \partial \eta_1^T} \right\},
\]

(7.2.15)

where the chain rule also yields

\[
\frac{d^2 \ell}{d (\eta_1)^2} = \frac{d^2 \ell}{d (\eta_1)^2} + \frac{d^2 \ell}{d \eta_1^2} \frac{d^2 \theta_1^2}{d \eta_1^2} \frac{d^2 \theta_1^2}{d \eta_1^2} + \left\{ \frac{d \ell}{d \theta_1} \frac{d^2 \theta_1^2}{d \eta_1^2} \right\}, \quad j \neq k,
\]

(7.2.16)

conforming with equations (18.7)–(18.9) in Yee (2015, Section 18.2.1).

After taking expections accordingly, as we wish to derive the EIMs, the above terms in braces \{ \cdots \} vanish, so that the diagonal and off–diagonal elements of the EIMs are given by

\[
-\mathbb{E} \left[ \frac{d^2 \ell}{d (\eta_1)^2} \right] = -\mathbb{E} \left[ \frac{d^2 \ell}{d (\eta_1)^2} \right] + \left( \frac{d \theta_1^k}{d \eta_1} \right)^2, \quad k = 1, 2,
\]

(7.2.18)

However, while (7.2.15)–(7.2.18) are well–known expressions concerning the second–order derivatives of vector–valued functions, it is the individual components on the right side of (7.2.18) that are to be affected, in addition to the score vector, by the 2–parameter dependency of the linear predictors stated in (7.2.5). For instance, consider the case when \( \eta_1 = G_1(\theta_1) \), and \( \eta_2 = G_2(\theta_1, \theta_2) \), with \( \eta_1, \theta_2 = (\eta_1, \eta_2) \), producing a composite log–likelihood \( \ell (\theta_1^k, \theta_2^k; y) \). From Table 7.2.1, and accounting for \( \theta_2^k = \theta_2^k(\theta_1^k) \), we have that

\[
\frac{d \ell}{d \theta_1^k} = \frac{\partial \ell}{\partial \theta_1^k} + \frac{\partial \ell}{\partial \theta_2^k} \cdot \frac{\partial \theta_2^k}{\partial \theta_1^k}.
\]

(7.2.19)
Moreover, the second derivative is

$$\frac{\partial^2 \ell}{\partial (\theta_2^2)^2} = \frac{\partial^2 \ell}{\partial \theta_1^2 \partial \theta_2^2} \cdot \frac{\partial \theta_1^2}{\partial \theta_2^2} \cdot \frac{\partial \theta_2^2}{\partial \theta_1^2} + \frac{\partial^2 \ell}{\partial \theta_2^2^2} \cdot \frac{\partial \theta_2^2}{\partial \theta_1^2} \cdot \frac{\partial \theta_1^2}{\partial \theta_2^2},$$

where the required expressions have been derived using the chain rule, resulting in:

$$\frac{\partial}{\partial \theta_1^2} \left( \frac{\partial \ell}{\partial \theta_2^2} \right) = \frac{\partial \theta_2^2}{\partial \theta_1^2} \cdot \frac{\partial \theta_1^2}{\partial \theta_2^2} \cdot \frac{\partial \theta_1^2}{\partial \theta_1^2}, \quad \text{and} \quad \frac{\partial}{\partial \theta_2^2} \left( \frac{\partial \ell}{\partial \theta_1^2} \right) = \frac{\partial \theta_1^2}{\partial \theta_2^2} \cdot \frac{\partial \theta_2^2}{\partial \theta_1^2} \cdot \frac{\partial \theta_2^2}{\partial \theta_2^2}.$$
Table 7.2.3. New complementary expressions for 2-parameter VGLMs to compute the off–diagonal elements of the EIMs, viz. (7.2.18), for the three versions of the linear predictors (7.2.5).

<table>
<thead>
<tr>
<th>( \eta_{G_1, G_2} = (\eta_{G_1}, \eta_{G_2}) )</th>
<th>Composite-( \ell )</th>
<th>Complementary expressions for ( d^2 \ell/d\eta_{G_1}^2 d\eta_{G_2}^2 ), ( d^2 \ell/d\eta_{G_1}^2 d\eta_{G_2}^2 ) in (7.2.18)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \eta_{G_1} = G_1(\theta_1, \theta_2) ), ( \ell(\theta_{G_1}^1(\theta_{G_2}^1, \theta_{G_2}^2); y) )</td>
<td>( \frac{d^2 \ell}{d\eta_{G_1}^2 d\eta_{G_2}^2} = \frac{\partial^2 \ell}{\partial \theta_{G_1}^1 \partial \theta_{G_2}^1} + \frac{\partial^2 \ell}{\partial (\theta_{G_2}^1)^2} \cdot \frac{\partial \theta_{G_2}^1}{\partial \theta_{G_2}^1}, )</td>
<td>( \frac{d^2 \ell}{d\eta_{G_1}^2 d\eta_{G_2}^2} = \frac{\partial^2 \ell}{\partial \theta_{G_1}^1 \partial \theta_{G_2}^1} + \frac{\partial^2 \ell}{\partial (\theta_{G_2}^1)^2} \cdot \frac{\partial \theta_{G_2}^1}{\partial \theta_{G_2}^1}, )</td>
</tr>
<tr>
<td>( \eta_{G_2} = G_2(\theta_2) )</td>
<td>( \frac{d^2 \ell}{d\eta_{G_2}^2 d\eta_{G_2}^2} = \frac{\partial^2 \ell}{\partial \theta_{G_2}^1 \partial \theta_{G_2}^1} + \frac{\partial^2 \ell}{\partial (\theta_{G_2}^1)^2} \cdot \frac{\partial \theta_{G_2}^1}{\partial \theta_{G_2}^1}, )</td>
<td>( \frac{d^2 \ell}{d\eta_{G_2}^2 d\eta_{G_2}^2} = \frac{\partial^2 \ell}{\partial \theta_{G_2}^1 \partial \theta_{G_2}^1} + \frac{\partial^2 \ell}{\partial (\theta_{G_2}^1)^2} \cdot \frac{\partial \theta_{G_2}^1}{\partial \theta_{G_2}^1}, )</td>
</tr>
</tbody>
</table>

\( \dagger \) Note that \( \frac{\partial^2 \ell}{\partial \theta_{G_1}^1 \partial \theta_{G_2}^2} \) implies \( \frac{d^2 \ell}{d\eta_{G_1}^2 d\eta_{G_2}^2} = \frac{d^2 \ell}{d\eta_{G_2}^2 d\eta_{G_2}^2}. \)

\[
\frac{\partial}{\partial \theta_{G_1}^1} \left[ \frac{\partial \ell}{\partial \theta_{G_1}^1} \cdot \frac{\partial^2 \ell}{\partial \theta_{G_2}^2} \cdot \frac{\partial \theta_{G_2}^2}{\partial \theta_{G_1}^1} \right] = \frac{\partial \ell}{\partial \theta_{G_1}^1} \cdot \frac{\partial^2 \theta_{G_2}^2}{\partial (\theta_{G_1}^1)^2} + \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \cdot \left( \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} + \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \right) \cdot \left( \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \right)^2 \tag{7.2.22}
\]

and replacing (7.2.21)–(7.2.22) in (7.2.20) and assuming \( \frac{\partial^2 \ell}{\partial \theta_{G_1}^1 \partial \theta_{G_1}^2} = \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1}, \) gives

\[
\frac{d^2 \ell}{d (\theta_{G_1}^1)^2} = \frac{\partial^2 \ell}{\partial (\theta_{G_1}^1)^2} + \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \cdot \frac{\partial \theta_{G_2}^2}{\partial \theta_{G_1}^1} + \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \cdot \frac{\partial \theta_{G_2}^2}{\partial \theta_{G_1}^1} \cdot \left( \frac{\partial^2 \ell}{\partial \theta_{G_2}^2 \partial \theta_{G_1}^1} \right)^2 \tag{7.2.23}
\]
7.2. On estimating VGLMs with 2-parameter link functions

as required for this example. Further, \( \frac{d^2 \ell}{d\theta_1^2 d\theta_2^2} \) in (7.2.18) has been derived in a similar manner, while \( \frac{\partial \theta_1}{\partial \theta_2^2} \) can be readily obtained from (7.2.13), \( j, k = 1, 2 \). The list of new complementary expressions to compute the EIMs for 2-parameter VGLMs (viz. (7.2.18)) is given in Tables 7.2.2 and 7.2.3.

Computationally, its implementation requires familiarity with basic aspects of VGLM family functions and its coding. Chapter 18 of Yee (2015) provides a comprehensive guide on what VGLM family functions consist of as well as directions for writing VGLM–links.

7.2.2 Implementing the NB canonical link

As a specific example, the NB distribution with a canonical link has now been correctly implemented in VGAM. Ordinarily, NB regressions involve two 1–parameter linear predictors given by

\[
\eta_1(\mu; x) = \log \mu, \quad \eta_1(k; x) = \log k, \quad (7.2.24)
\]

which can be estimated via the family function \texttt{VGAM::negbinomial()} . When the 2–parameter NB–canonical link (\( \eta_1 \) below) is considered, the linear predictors become (viz. (7.2.4)):

\[
\eta_1(\mu, k; x) = \beta_1^T x = \log \frac{\mu}{\mu + k}, \quad (7.2.25)
\]

\[
\eta_2(k; x) = \beta_2^T x = \log k.
\]

If \( k \) is known, then \( \eta_1 \) can be fitted as a GLM, and this NB variant, called NB–C1–H, can be estimated with \texttt{VGAM::negbinomial.size()} . However, the usual case is when \( k \) is to be estimated, giving place to the so–called NB-C2-2 model, and consequently a few modifications to the VGLM–framework based on the methodology given in Section 7.2.1 are required, as follows:

1) To implement the VGLM–family function to estimate NB-C2-2, and
2) to implement the corresponding 2–parameter VGLM–link.

To date, both tasks are completed. First, \texttt{VGAM::negbinomial()} has been adapted, such that it now accommodates the modified score vector and EIMs, as depicted in Tables 7.2.1 and 7.2.2, thus satisfying 1); and secondly, we have modified the VGLM–link \texttt{nbcanlink()} in \texttt{VGAM} to handle the 2–parameter NB canonical link, i.e., \( \eta_1 \) in (7.2.25), hence satisfying 2). Both enhanced functions are available in \texttt{VGAM} \( \geq 1.0.6 \).

In this section we give the details of both implementations, with emphasis on the modified versions of the score vector and the EIMs currently handled by...
VGAM::negbinomial(). Besides, we compare the performance of the preceding version of VGAM::nbcanlink(), included in VGAM 1.0-0 (Yee, 2015), with the current VGAM::nbcanlink() (from VGAM ≥ 1.0-6) and show that the former was incorrect. While fitting the NB distribution with a canonical link is more of theoretical interest than a practical method since $\eta_1(\theta_1, \theta_2; x) < 0$ (an ordinary VGLM linear predictor ought to be unbounded), we take this NB variant as an example to demonstrate the correctness of the proposed methodology.

7.2.2.1 Adapting negbinomial() to estimate NB–C$_2$–2.

We start with the derivation of the modified score vector and the modified expressions for the EIMs of NB–C$_2$–2 that have been integrated into negbinomial().

Let $Y \sim \text{NB}(\mu, k; x)$, and let $y = \{y_1, \ldots, y_n\}$ be a random sample of such. Consider the associated VGLM, say $F(y|x; B)$ with covariates $x$, and two linear predictors $\eta_1(x)$ and $\eta_2(x)$. Regardless, the functional form of $\eta_1$ and $\eta_2$, the $i$th contribution to the log–likelihood, $\ell(\eta_1, \eta_2; y, x)$, is given by

$$\ell_i(\mu_i, k_i; y_i) = \log \Gamma(y_i + k_i) - \log \Gamma(y_i + 1) - \log \Gamma(k_i) + y_i \log \left( \frac{\mu_i}{\mu_i + k_i} \right) + k_i \log \left( \frac{k_i}{\mu_i + k_i} \right).$$

With the ordinary linear predictors of (7.2.24), one has

$$\frac{d\ell_i}{dk_i} = \frac{\partial\ell_i}{\partial\mu} \cdot \frac{\partial\mu}{\partial k} + \frac{\partial\ell_i}{\partial k} = \psi(y_i + k_i) - \psi(k_i) + \log \left( \frac{k_i}{k_i + \mu_i} \right) - \frac{y_i - \mu_i}{k_i + \mu_i}. \quad (7.2.27)$$

However, with two linear predictors as in (7.2.25), the relationship

$$\mu_i = \mu_i(k_i) = \frac{k}{e^m - 1}$$

holds, implying from Table 7.2.1 that (the subscript $i$ is dropped for simplicity)

$$\frac{d\ell}{dk} = \frac{\partial\ell}{\partial\mu} \cdot \frac{\partial\mu}{\partial k} + \frac{\partial\ell}{\partial k}. \quad (7.2.28)$$

The partial derivatives here are directly computed from (7.2.26), resulting in:

$$\frac{\partial\mu}{\partial k} = \frac{1}{e^m - 1} = \frac{\mu}{k}, \quad (7.2.29)$$

$$\frac{\partial\ell}{\partial\mu} = \frac{y}{\mu} - \frac{y}{\mu + k} - \frac{k}{\mu + k},$$

$$\frac{\partial\ell}{\partial k} = \psi(y + k) - \psi(k) - \frac{y}{\mu + k} + (1 + \log k) - \left[ \frac{k}{\mu + k} + \log(\mu + k) \right].$$
and replacing such in (7.2.28) gives:

\[
\frac{d}{dk} \ell = \left( y - \frac{y}{\mu + k} - \frac{k}{\mu + k} \right) \cdot \frac{\mu}{k} + \psi(y + k) - \psi(k) - \psi'(y + k) - \psi'(k) + \frac{1}{\mu + k} \left( \frac{y}{\mu + k} + \frac{k}{\mu + k} \right) + 1 + \log \left( \frac{k}{\mu + k} \right)
\]

which is (7.2.27) without its last term. Particularly, \( \frac{d}{d\mu} \ell = \frac{\partial}{\partial \mu} \ell \) is given in (7.2.29), completing the score vector.

To derive the EIM of the NB-C2-2 model, we must consider (7.2.18) along with the results from Table 7.2.2. Note that the EIM is non–diagonal, where the adjusted non–diagonal component is given by

\[
\frac{d^2}{d\mu dk} = \frac{\partial}{\partial \mu} - \log(\mu + k) = \frac{-1}{\mu + k} \Rightarrow -E \left[ \frac{d^2 \ell}{d\mu dk} \right] = \frac{1}{\mu + k}.
\]

Moreover, while the 1–1 element remains as with the ordinary linear predictors, the 2–2 element also needs adjustment:

\[
-\mathbb{E} \left[ \frac{d^2 \ell}{dk^2} \right] = W_{kk} + \frac{\mu}{k(\mu + k)} = \psi'(k) - \mathbb{E}[\psi'(Y + k)],
\]

where \( W_{kk} \) is the usual 2–2 element involving trigamma functions corresponding to 1-parameter links such as the logarithm. Putting everything together, with the default (7.2.25) and following from (7.2.18), the working weight matrix has the following expression:
\[
\begin{pmatrix}
\frac{1}{\mu} - \frac{1}{\mu + k} & \frac{1}{\mu + k} \\
\frac{1}{\mu + k} & \psi'(k) - E[\psi'(Y + k)]
\end{pmatrix}
\circ
\begin{pmatrix}
\left( \frac{d\mu}{d\eta_1} \right)^2 & \frac{d\mu}{d\eta_1} \frac{dk}{d\eta_2} \\
\frac{d\mu}{d\eta_1} \frac{dk}{d\eta_2} & \left( \frac{dk}{d\eta_2} \right)^2
\end{pmatrix}
\]

The functions \( d\mu/d\eta_1 \) and \( dk/d\eta_2 \) are internally computed by \texttt{VGAM::dtheta.deta()} at each Fisher scoring iteration. The list of functions that make use of a VGLM–link, e.g., to compute the derivatives such as \texttt{VGAM::dtheta.deta()} is given in Yee (2015, Table 18.2).

### 7.2.2.2 Computational details of \texttt{nbcanlink()}

In this section we briefly review the second subject related to adapting the VGLM–framework to estimate the NB-C\(_2\)-2 model: a VGLM–link to handle its canonical link. This work is currently performed by an enhanced \texttt{nbcanlink()} which computationally describes \( \eta_1(\theta_1, \theta_2; x) \) in (7.2.25), with \( \theta_2 = k \) unknown but directly estimable from (7.2.25), since \( \theta_2 = \exp \eta_2 \), at every Fisher scoring iteration.

The arguments handled by this link function are the following. The complete list of compulsory arguments ought to be considered when writing VGLM–links can be found in Yee (2015, Table 18.1).

```r
> args(nbcanlink)
function (theta, size = NULL, wrt.param = NULL, bvalue = NULL, inverse = FALSE, deriv = 0, short = TRUE, tag = FALSE)
NULL
```

From this output, the attention is focussed on the following three arguments due to its central connection with the new methodology proposed in this chapter:

1. **wrt.param**, a positive integer that specifies which parameter the derivatives should be computed with respect to and returned. It is either 1 or 2, where 1 indicates ‘with respect to the first parameter’, \( \mu \), and 2 for ‘with respect to the second parameter’, i.e., \( k \).
2. **inverse**. Logical. For the default (\( \text{inverse} = \text{FALSE} \)), the \( \eta \)–vector (7.2.25) is returned, else, the inverse–link values, \( \theta \), computed from the inverse–system (7.2.13) are returned. Conveniently, this argument can be used together with the other two arguments, as shown below.
3. **deriv**, either 0, 1, or 2 to indicate the order of the derivative to be computed, as follows:
When \( \text{inverse} = \text{TRUE} \) the dependent parameters are \( \mu \) and \( k \), by virtue of (7.2.13). When, in addition, \( \text{wrt.param} = 1 \), thus referring to \( \eta_1 \), we have:

(i) For \( \text{deriv} = 0 \), the new \text{nbcanlink()} returns the inverse function (cf. (7.2.13))

\[
\mu(\eta_1, k) = \frac{k}{e^{-\eta_1} - 1}.
\]  

(7.2.33)

(ii) For \( \text{deriv} = 1 \), then

\[
\frac{d\mu}{d\eta_1} = \frac{\mu \cdot (\mu + k)}{k},
\]  

which is to be used in (7.2.12) and (7.2.18), is returned (assuming \text{type.fitted} = "mean").

(iii) Likewise, when \( \text{deriv} = 2 \), then

\[
\frac{d^2 \mu}{d(\eta_1)^2} = \frac{k(\mu + k)(2\mu + k)}{k^2}.
\]  

(7.2.35)

When \( \text{inverse} = \text{TRUE} \) and \( \text{wrt.param} = 2 \), thus referring to \( \eta_2 \), then:

(i) For \( \text{deriv} = 0 \), it returns NULL, since the inverse \( k = \exp \eta_2 \) is directly computed with \text{loglink}(..., \text{inverse} = \text{TRUE}, ...).

(ii) For \( \text{deriv} = 1 \), \text{nbcanlink()} returns

\[
\frac{dk}{d\eta_2} = -(\mu + k).
\]  

(7.2.36)

(iii) If \( \text{deriv} = 2 \), then

\[
\frac{d^2 k}{d(\eta_2)^2} = \mu + k.
\]  

(7.2.37)

When \( \text{inverse} = \text{FALSE} \), then \text{nbcanlink()} returns:

(i) The linear predictor \( \eta_1 = \log \frac{\mu}{\mu + k} \), when \( \text{deriv} = 0 \),

(ii) The vector \( \frac{d\eta_1}{d\mu} = \frac{k}{\mu(\mu + k)} = \frac{1}{d\mu/d\eta_1} \), if \( \text{wrt.param} = 1 \), or \( \frac{d\eta_1}{dk} = -\frac{1}{\mu + k} \), if \( \text{wrt.param} = 2 \), when \( \text{deriv} = 1 \), and

(iii) either the vector \( \frac{d^2 \eta_1}{d\mu^2} = \frac{-k(2\mu + k)}{[\mu \cdot (\mu + k)]^2} \), if \( \text{wrt.param} = 1 \),

\[
\frac{d^2 \eta_1}{dk^2} = \frac{1}{(\mu + k)^2}, \text{if} \ \text{wrt.param} = 2, \text{when} \ \text{deriv} = 2.
\]
As an example, the following R code shows an output produced with `nbcanlink()`, in VGAM 1.0-6, with `deriv = 1`, `wrt.param = 1` and `inverse = TRUE`. Here, \( \frac{d\mu}{d\eta_1} \) is returned.

```r
> size <- rep(3, 5)  # This is \( k = c(3, 3, 3, 3, 3) \), or simply \( k = 3 \).
> wrt.param <- 1
> inverse <- TRUE
> deriv <- 1
> theta <- 1:5  # This is \( \mu = c(1, 2, 3, 4, 5) \)
> t(nbcanlink(theta, wrt.param = wrt.param, deriv = deriv,
inverse = inverse, size = size))
```

```
theta 1.33 3.33 6 9.33 13.3
```

We conclude this section with an example using simulated data that compares the performance of the current `nbcanlink()` with the old version from VGAM 1.0-0 (Yee, 2015). For clarity, we have re-labelled the latter link as `nbcanlink.old()`. The difference is that `nbcanlink()` has been implemented following the methodology proposed in Section 7.2.1, while `nbcanlink.old()` relies on (7.2.9)–(7.2.10) ignoring the crucial expressions (7.2.11)–(7.2.12). This example aims to demonstrate the numerical instability of Fisher scoring when `nbcanlink.old()` is utilized, which has been referred in the literature (see, e.g., Hilbe, 2011).

The data \( (n = 200) \) is generated by making use of `stats:rnbinom()` involving two independent sets of random deviates from the NB distribution allocated to two responses, \( Y_1 \sim \text{NB}(\mu_1, k_1; x) \), and \( Y_2 \sim \text{NB}(\mu_2, k_2; x) \), for covariates \( x = (x_1, x_2)^T \).

The associated VGLMs are, say, \( F_1(y_1; \eta_{11}(x), \eta_{12}(x)) \), and \( F_2(y_2; \eta_{21}(x), \eta_{22}(x)) \), and, in the following, VGAM::negbinomial() will be used to estimate such. The default linear predictors handled by this VGLMff, per response \( \kappa, \kappa = 1, 2 \), are

\[
\eta_{\kappa,1}(x) = \log \frac{\mu_\kappa}{\mu_\kappa + k_\kappa}, \quad \text{and} \quad \eta_{\kappa,2} = \log k_\kappa,
\]

and, essentially, `stats:rnbinom()` requires the inputs of \( \mu_\kappa \), and \( k_\kappa \), that are also generated, provided \( \eta_{\kappa,1}(x) < 0 \), as follows:

```r
n <- 200; set.seed(123)
### Two artificial covariates, 'x2' and 'x3' from a uniform distribution.
ndata <- data.frame(x2 = runif(n), x3 = runif(n))
### Generate eta_{1, 1}, eta_{1, 2}, eta_{2, 1}, and eta_{2, 2}
ndata <- transform(ndata, eta11 = -0.2 - 0 * x2,  # eta_{11}, eta_{12} < 0 NOT needed
                   eta12 = 1.0,  # eta_{12} < 0 NOT needed
```
7.2. On estimating VGLMs with 2–parameter link functions

\[
\begin{align*}
\eta_{11} &= -0.2 - 0.1 \times x_2 - 0.5 \times x_3, \\
\eta_{22} &= 1.0 + 1 \times x_2 \quad \# \eta_{22} < 0 \text{ NOT needed}
\end{align*}
\]

### Generating the vectors of size parameters 'k_1', 'k_2'
ndata <- transform(ndata, size1 = exp(eta12), size2 = exp(eta22))
### With 'inverse = TRUE' we generate the vectors of means 'mu_1', 'mu_2'
ndata <- transform(ndata, 
  mu1 = nbcanlink(eta11, size = size1, inverse = TRUE),
  mu2 = nbcanlink(eta21, size = size2, inverse = TRUE))
### The responses 'y_1' and 'y_2' are obtained from rbinom().
ndata <- transform(ndata, 
  y1 = rnbinom(n, mu = mu1, size = size1),
  y2 = rnbinom(n, mu = mu2, size = size2))

Note that the true linear predictors are:

\[
\begin{align*}
\eta_{11}(x) &= \log \frac{\mu_1}{\mu_1 + k_1} = -0.2 - 0.0 \times x_2, \quad \eta_{12}(x) = \log k_1 = 1.0, \quad (7.2.39) \\
\eta_{21}(x) &= \log \frac{\mu_2}{\mu_2 + k_2} = -0.2 - 0.1 \times x_2 - 0.5 \times x_3, \quad (7.2.40) \\
\eta_{22}(x) &= \log k_2 = 1.0 + 1.0 \times x_2.
\end{align*}
\]

Firstly, the results after fitting a one–response VGLM with negbinomial() and the old–version of nbcanlink() are shown. Note from the warning message that Fisher scoring wriggles towards the solution. Further advise on other potential issues are given in the corresponding documentation file in VGAM 1.0-0. We have set trace = TRUE to show how fragile and slow this statistical model was in previous versions, as well as criterion = "coefficients", as evidence on the observed numerical instability in the estimated coefficients. Usually, about 6–8 iterations are needed for Fisher scoring. This fit is called fit.using.old.link. By default zero = "size", which means that \( \eta_1 = \log k \) is intercept–only, as pointed in (7.2.38).

> fit.using.old.link <- vglm(y1 ~ x2 + x3, negbinomial(lmu = "nbcanlink.old"),
  data = ndata, criterion = "coefficients", trace = TRUE) # VGAM 1.0-0
VGLM  linear loop 1 :  coefficients =
  -0.1500585337,  1.0078820209, -0.0050649846, -0.0981940482
VGLM  linear loop 2 :  coefficients =
  -0.14664165577,  1.01689551244, -0.00021366482, -0.11196022535
VGLM  linear loop 3 :  coefficients =
  -0.1553177280,  1.0138162841, -0.0041721039, -0.1026598602
VGLM  linear loop 4 :  coefficients =
  -0.1516739258,  1.0194801008,  0.0008733857, -0.1169892069
VGLM  linear loop 5 :  coefficients =
  -0.152096224,  1.017416781, -0.004402796, -0.102176060
VGLM  linear loop 6 :  coefficients =

> fit.using.old.link <- vglm(y1 ~ x2 + x3, negbinomial(lmu = "nbcanlink.old"),
  data = ndata, criterion = "coefficients", trace = TRUE) # VGAM 1.0-0
Chapter 7. Miscellaneous VGLM Topics

-0.15238118275, 1.01946304707, 0.00065671187, -0.11645008044
VGLM linear loop 7: coefficients =
-0.156126810, 1.017661343, -0.004443246, -0.102083352
VGLM linear loop 8: coefficients =
-0.15240881642, 1.01946305057, 0.0006147409, -0.1163463004
VGLM linear loop 9: coefficients =
-0.15612778818, 1.017664414, -0.004448254, -0.1020823396
VGLM linear loop 10: coefficients =
-0.1524093592, 1.0196629332, 0.00061466328, -0.11634615535
VGLM linear loop 11: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 12: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522
VGLM linear loop 13: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 14: coefficients =
-0.1524093592, 1.0196629332, 0.00061466328, -0.11634615535
VGLM linear loop 15: coefficients =
-0.15612778818, 1.017664414, -0.004448254, -0.1020823396
VGLM linear loop 16: coefficients =
-0.1524093592, 1.0196629332, 0.00061466328, -0.11634615535
VGLM linear loop 17: coefficients =
-0.15612778818, 1.017664414, -0.004448254, -0.1020823396
VGLM linear loop 18: coefficients =
-0.1524093592, 1.0196629332, 0.00061466328, -0.11634615535
VGLM linear loop 19: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 20: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522
VGLM linear loop 21: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 22: coefficients =
-0.1524093592, 1.0196629332, 0.00061466328, -0.11634615535
VGLM linear loop 23: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 24: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522
VGLM linear loop 25: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 26: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522
VGLM linear loop 27: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 28: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522
VGLM linear loop 29: coefficients =
-0.1561276906, 1.0176689742, -0.004449051, -0.1020822105
VGLM linear loop 30: coefficients =
-0.1524090375768, 1.01966293266, 0.000614666328, -0.11634615522

Warning in vglm.fitter(x = x, y = y, w = w, offset = offset, Xm2 = Xm2, : convergence not obtained in 30 IRLS iterations
Now, we show the results after fitting the same VGLM to the data from \( Y_1 \) using the actual \texttt{nbcanlink()} that has been implemented based on the methodology proposed in this chapter. This fit is called \texttt{fit.using.new.link}, where, similar to \texttt{fit.using.old.link}, \( \eta_1 = \log k \) is intercept-only.

\begin{verbatim}
> fit.using.new.link <- vglm(y1 ~ x2 + x3, negbinomial(lmu = "nbcanlink"),
    data = ndata, criterion = "coefficients", trace = TRUE) # VGAM >= 1.0-6
VGLM linear loop 1: coefficients =
  -0.1379396277, 0.8781495191, -0.0032297081, -0.0631854807
VGLM linear loop 2: coefficients =
  -0.1521788984, 0.9929369854, -0.0042838701, -0.0937426820
VGLM linear loop 3: coefficients =
  -0.1560736060, 1.0188226040, -0.0045273831, -0.1016876848
VGLM linear loop 4: coefficients =
  -0.1562669039, 1.0199527129, -0.0045383768, -0.1020825717
VGLM linear loop 5: coefficients =
  -0.1562657253, 1.0199440725, -0.0045383584, -0.1020825731
VGLM linear loop 6: coefficients =
  -0.1562657389, 1.0199441661, -0.0045383587, -0.1020825809
The trace output looks natural and correct. Incidentally, we illustrate the performance of \texttt{negbinomial()} using \texttt{nbcanlink()} when fitting a 2–response VGLM. Here, we have particularly set \texttt{zero = NULL} allowing both linear predictors to be regressed on \( x_2 \) and \( x_3 \). The estimated coefficients are given post estimation (cf. (7.2.39)–(7.2.40)).

\begin{verbatim}
> fit2.using.new.link <- vglm(cbind(y1, y2) ~ x2 + x3, negbinomial(lmu = "nbcanlink", zero = NULL),
    data = ndata, criterion = "loglikelihood") # VGAM >= 1.0-6.
> t(coef(fit2.using.new.link, matrix = TRUE))

## The estimated coefficients
(Intercept) x2   x3
nbcanlink(mu1, mu1(size1)) -0.15446 0.022658 -0.14205
loglink(size1) 1.00619 -0.154232 0.21012
nbcanlink(mu2, mu2(size2)) -0.21734 -0.086665 -0.31045
loglink(size2) 1.03110 1.068344 -0.40971

> fit2.using.new.link@iter

## Number of IRLS iterations for convergence
[1] 9
\end{verbatim}

The labelling "\texttt{nbcanlink(mu1, mu1(size1))}" and "\texttt{nbcanlink(mu2, mu2(size2))}" conforms with \texttt{VGAM \geq 1.0-6}, and conveys the idea behind this VGLM–link extension.

### 7.3 On mean modelling of several 1–parameter distributions

In this section a few more central topics emerged from the work towards modelling of VGLMs are considered. Unlike the results of Section 7.2 with 2–parameter distributions, the following concentrates on 1–parameter distributions.

GLMs (Nelder and Wedderburn, 1972) and GAMs (Hastie and Tibshirani, 1986, 1990) are well–known popular statistical frameworks handling different data types by connecting the mean–function of the 1–parameter exponential family. Since their introduction, several GLM/GAM extensions have been proposed, for instance, GAMs for
### Table 7.3.1. New link functions for the mean of some discrete distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>( \theta )</th>
<th>Range of ( \theta )</th>
<th>Mean ( \mu )</th>
<th>Link function ([\eta(\theta)])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borel–Tanner</td>
<td>( a )</td>
<td>(0, 1)</td>
<td>( Q/(1 - \theta) )</td>
<td>( \log\text{link}) ( Q^{-1} - \theta Q^{-1} )</td>
</tr>
<tr>
<td>Geometric</td>
<td>( p )</td>
<td>(0, 1)</td>
<td>( (1 - \theta)/\theta )</td>
<td>(-\log\text{it}(\theta))</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>( s ) (\dagger)</td>
<td>(0, 1)</td>
<td>( \theta ) ( (1 - \theta) [-\log(1 - \theta)] )</td>
<td>( \log\text{it}(\theta) - \text{clog\log\text{it}}(\theta))</td>
</tr>
<tr>
<td>Positive Poisson</td>
<td>( \lambda )</td>
<td>(0, (\infty))</td>
<td>( \theta ) ( \frac{1}{1 - e^{-\theta}} )</td>
<td>(-\log\text{link}) ( \theta^{-1} - \theta^{-1} e^{-\theta} )</td>
</tr>
<tr>
<td>Yule–Simon</td>
<td>( \rho ) (\dagger)</td>
<td>(0, (\infty))</td>
<td>( \theta ) ( \frac{\theta}{\theta - 1}, \theta &gt; 1 )</td>
<td>(-\log\text{link}) ( 1 - \theta^{-1} )</td>
</tr>
<tr>
<td>Zeta (Zipf)</td>
<td>( s ) (\dagger)</td>
<td>(0, (\infty))</td>
<td>( \zeta(\theta)/\zeta(\theta + 1), \theta &gt; 1 )</td>
<td>( \log\text{link}) ( \zeta(\theta)/\zeta(\theta + 1) )</td>
</tr>
</tbody>
</table>

\(\dagger\) The density and the moments of the Zipf distribution conforms with the VGLM–family function \textit{zetaff()} in \textbf{VGAM}. Here, \( \zeta \) is the Riemann zeta function.

\(\dagger\) These are ‘shape’ parameters.

Developed under the VGLM/VGAM approach, we return full circle by proposing new link–functions for the mean of several 1–parameter distributions residing outside the exponential family. This work springs from the ability of VGLMs/VGAMs to handle parameter–links \( g_j(\theta_j) \) as smooth and monotonic functions of the parameters, for example the mean of the logarithmic distribution with parameter \( 0 < s < 1 \), is given by

\[
\mathbb{E}_\theta(Y) = \mu_Y = \frac{1}{-\log(1 - s)} \cdot \frac{s}{1 - s}, \tag{7.3.1}
\]

for \( Y \sim \text{Logarithmic}(s) \), rather than single transformations of such, via ordinary links, e.g., \( \log s \). Within \textbf{VGAM} only a few family functions have been implemented under this approach, including \textit{Betabinomial()} which fits a beta–binomial distribution by Fisher scoring where the mean and the correlation coefficient are estimated, or \textit{betaff()} to estimate the mean and the precision parameters of the Beta distribution.

The main motivation for this work is two–fold: (a) the mean function is most easily interpreted, and (b) event–rate parameters of the form \( \mu_i = \lambda_i \cdot t_i \), can be readily handled by incorporating ‘time’ as an \textit{offset} when the logarithm is taken. Typically, within \textbf{GAMLSS} and \textbf{GAMs}, problems of this nature might be approached via Poisson regression with linear predictor

\[
\log \mu_i = \beta^T x_i + \log t_i
\]
Table 7.3.2. New VGLM–links for the mean of some continuous 1–parameter distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>θ</th>
<th>Range of θ</th>
<th>Support</th>
<th>Mean μ</th>
<th>Link function [η(θ)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential †</td>
<td>λ</td>
<td>(0, ∞)</td>
<td>(A, ∞)</td>
<td>A + θ⁻¹</td>
<td>loglink(A + θ⁻¹)</td>
</tr>
<tr>
<td>Inverse–χ² df</td>
<td>df</td>
<td>[0, ∞)</td>
<td>(0, ∞)</td>
<td>1 / (θ - 2), θ &gt; 2</td>
<td>-loglink(θ - 2)</td>
</tr>
<tr>
<td>Maxwell §</td>
<td>a</td>
<td>(0, ∞)</td>
<td>(0, ∞)</td>
<td>a⁻¹/2 √(8/π)</td>
<td>θ - 2</td>
</tr>
<tr>
<td>Rayleigh</td>
<td></td>
<td></td>
<td>b</td>
<td>(0, ∞)</td>
<td>(0, ∞)</td>
</tr>
<tr>
<td>Topp–Leone §</td>
<td>s</td>
<td>(0, 1)</td>
<td>(0, 1)</td>
<td>1 - 4θΓ(1+θ)² / Γ(2+2θ)</td>
<td>logitlink(μ(θ) / κ₃)</td>
</tr>
</tbody>
</table>

† A is a location parameter (fixed) and λ is a rate.
§ κ₁ := 3 / 2 log 2 - log Γ(0.5) ≈ 0.4673.
∥ κ₂ := log Γ(0.5) - 1 / 2 log 2 ≈ 0.2258.
‡ κ₃ := sup₀<θ<₁ {1 - 4θΓ(1+θ)² / Γ(2+2θ)}

for a set of covariates xᵢ. However, the VGLM–framework is equipped with 150+ distributional choices beyond the exponential family such as the Yule–Simon distribution for rare events in a long–period time, where data in the tails are sparsely sampled but very likely influencing on the estimates and SEs.

Specifically, this work proposes new link functions for the mean of 1–parameter distributions (say μᵧ), based on its log–transformation, giving place to amenable relationships between ordinary link functions. For instance, from (7.3.1), the implemented mean–link is called logffMeanlink(), based on:

\[
\log \mu_Y = \log \left( \frac{s}{1 - s} \right) - \log \left( -\log(1 - s) \right) = \logitlink(s) - \text{clogloglink}(s) = \text{logffMeanlink}(s),
\]

that is, the difference between the ordinary VGLM–links logitlink() and clogloglink(). A summary of the new mean–links are given in Table 7.3.1 (for discrete distributions) and Table 7.3.2 (for continuous distributions). These functions have been implemented in VGAMextra, named as in Table 7.3.3. The typical usage remains as with any VGLM–link, that is to assign the mean–link to the corresponding argument in the VGLMff. Following the above example, VGAM::logff() is the VGLMff to estimate the logarithmic distribution, with s being the shape parameter. Then, for an ordinary VGLM–fitting call, one sets:

```r
fit.log <- vglm(y ~ x, logff(lshape = logffMeanlink), data = logdata)
```
Table 7.3.3. Mean–links currently available in VGAMextra

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Name in VGAMextra</th>
<th>Distribution</th>
<th>Name in VGAMextra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Borel–Tanner</td>
<td>borel.tannerMeanlink()</td>
<td>Exponential</td>
<td>expMeanlink()</td>
</tr>
<tr>
<td>Geometric</td>
<td>geometricffMeanlink()</td>
<td>Inverse–χ²</td>
<td>inv.chisqMeanlink()</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>logffMeanlink()</td>
<td>Maxwell</td>
<td>maxwellMeanlink()</td>
</tr>
<tr>
<td>Pos. Poisson</td>
<td>posPoiMeanlink()</td>
<td>Rayleigh</td>
<td>rayleighMeanlink()</td>
</tr>
<tr>
<td>Yule–Simon</td>
<td>yulesimonMeanlink()</td>
<td>Topp–Leone</td>
<td>toppleMeanlink()</td>
</tr>
<tr>
<td>zeta (Zipf)</td>
<td>zetaffMeanlink()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

However, implementing VGLM–links is not an easy task. A number of compulsory arguments and conventions must be managed (see Table 18.1 in Yee (2015)), which, at the same time, require the following essential inputs that must be derived accordingly: (1) the first two derivatives, and (2) the inverse–link and its first two derivatives, that must be available at least numerically. In the following section we intend to illustrate this procedure for the logarithmic distribution towards its mean–link, logffMeanlink(). The remaining mean–links have been implemented in a similar fashion.

7.3.1 The logffMeanlink() VGLM–link

The logffMeanlink() is the mean–link of the logarithmic distribution defined as

\[
\text{logffMeanlink}(s) = \text{logitlink}(s) - \text{clogloglink}(s), \quad 0 < s < 1, \quad (7.3.3)
\]

which will be denoted as follows for simplicity: \( g(s) = \text{logffMeanlink}(s) \).

7.3.1.1 The first two derivatives of logffMeanlink()

The function \( g \) and its first two derivatives are directly computable from the links loglink() and clogloglink(), from VGAM. More precisely, for any \( 0 < s < 1 \),

\[
\frac{\partial^k g}{\partial s^k} = \text{logffMeanlink}(s, \text{deriv} = k, \text{inverse} = \text{FALSE}) \quad (7.3.4)
\]

\[
= \text{logitlink}(s, \text{deriv} = k, \text{inverse} = \text{FALSE}) - \text{clogloglink}(s, \text{deriv} = k, \text{inverse} = \text{FALSE}),
\]

for \( k = 0, 1, 2 \).
7.3. On mean modelling of several 1–parameter distributions

7.3.1.2 logffMeanlink(): Its inverse and its first two derivatives

To show how the inverse \( g^{-1} = \text{logffMeanlink}()^{-1} \) is derived, we start by analyzing the plot of \text{logffMeanlink}(), shown in Figure 7.3.1. This link is smooth, continuous and monotonic in \((0, 1)\), and grows rapidly as \( s \to 1 \). Moreover, from (7.3.2), this VGLM–link lacks a closed–form inverse, however it is one–to–one, meaning that for any \( \eta_i \in (0, \infty) \) there exists an unique \( s_i \in (0, 1) \) such that

\[
\text{logffMeanlink}(s_i) = g(s_i) = \eta_i. \tag{7.3.5}
\]

This property allows to obtain the “inverse” numerically by finding the root \( s^*_i \) of the function:

\[
g_{\eta_i}(s) = \text{logffMeanlink}(s_i) - \eta_i,
\]

for fixed \( \eta_i \), such that \( g_{\eta_i}(s^*_i) = 0 \) to machine precision. By means of \( g() \), \( g_{\eta_i}() \) is also smooth and monotonic, and its plot for some selected values of \( \eta_i \) is shown in Figure 7.3.2, indicating that these roots are real and consequently an iterative algorithm can be utilized to approximate them. Note especially that Figure 7.3.1 gives the plot when \( \eta_i = 0 \).

When implementing this, our data is indeed \( n \)–dimensional, say \( y_i = \{y_i, x_{i,p}\}, i = 1, \ldots, n \), and consequently we aim to find the roots \( s \in \mathbb{R}^n \) of \( g_{\eta_i}() \) for the fixed \( n \)–dimensional vector \( \eta = (\eta_1, \ldots, \eta_n)^T \) at each IRLS–iteration. To achieve this efficiently, \text{logffMeanlink}() handles two choices. These are: (a) \text{newtonRaphson.basic}(), available in \text{VGAMextra}, and (b) \text{VGAM::bisection.basic}(), two \text{vectorized} implementations of the well–known Newton–Raphson and bisection algorithms to approximate for

Figure 7.3.1. Plot of \text{logflink()} in \((0, 1)\).
Figure 7.3.2. Plot of $g_{\eta_i}(s)$ for: (a) $\eta_i = 0.5$, (b) $\eta_i = 1.5$, (c) $\eta_i = 3.0$, (d) $\eta_i = 6.0$.

less than a relative $\epsilon$–error the roots of real–valued functions in a given interval $(a, b)$. At each iteration $k$, the default of `newtonRaphson.basic()` in `logffMeanlink()` searches for $s^{(k+1)}$ such that

$$\frac{\|s^{(k+1)} - s^{(k)}\|}{\|s^{(k)}\|} < \epsilon,$$

for given $\epsilon$. The vectorization is a method to obtain greater efficiency in R rather than a `for()` loop. The choices (a)–(b) in `logffMeanlink()` are handled by `alg.roots`, in addition to the compulsory arguments required by VGLM–link functions. The complete list is the following (see Table 18.1 in Yee (2015)):

```r
> args(logffMeanlink)
```

```r
definition (theta, bvalue = NULL, alg.roots = c("Newton-Raphson",
    "bisection")[1], inverse = FALSE, deriv = 0, short = TRUE,
    tag = FALSE)
```

> NULL

With this approach the problem of determining the required inverse $g^{-1}$ is numerically solved at the cost, however, of replicating the process at every IRLS–iteration. The required inputs $\eta$ are internally generated and passed to `newtonRaphson.basic()`.

Regarding the first two derivatives of $g^{-1} = logffMeanlink()^{-1}$, these are simply computed by implicit differentiation. Indeed, for any suitable VGLM–link, particularly
7.3. On mean modelling of several 1–parameter distributions

Figure 7.3.3. Some 1–parameter probability VGLM–link functions, with $0 < \theta < 1$.

for \texttt{logffMeanlink()}, the linear predictor involved with \texttt{VGAM::logff()} is

\[
\eta = \text{logffMeanlink}(s) = \log \left( \frac{1}{-\log(1 - s(\eta))}, \frac{s(\eta)}{1 - s(\eta)} \right). \tag{7.3.6}
\]

where we aim to find $s'(\eta) = \partial s/\partial \eta$. However, recall that the vector $s$ should be found upon $\eta$, allowing to set the relation $s = s(\eta)$. Then, differentiating (7.3.6) produces:

\[
\frac{\partial}{\partial \eta} \eta = \frac{\partial}{\partial \eta} \log \left( \frac{1}{-\log(1 - s(\eta))}, \frac{s(\eta)}{1 - s(\eta)} \right) \iff \frac{1}{1} = \frac{\partial}{\partial \eta} \log \left( \frac{1}{-\log(1 - s(\eta))}, \frac{s(\eta)}{1 - s(\eta)} \right),
\]

and solving for $s' = \frac{\partial s}{\partial \eta}$ gives the first derivative, as:

\[
s'(\eta) = \frac{s(\eta)(1 - s(\eta)) \log(1 - s(\eta))}{s(\eta) + \log s(\eta)}. \tag{7.3.7}
\]

The second derivative of $g^{-1}$, $s'' = \frac{\partial^2 s}{\partial \eta^2}$, has a more complicated yet closed form, that is obtained by (once again) deriving (7.3.7) with respect to $\eta$. This produces

\[
\frac{\partial^2 s}{\partial \eta^2} = s' \left\{ \frac{(1 - 2s) \log(1 - s) - s}{s + \log(1 - s)} + \frac{s^2 \log(1 - s)}{[s + \log(1 - s)]^2} \right\}, \tag{7.3.8}
\]

where $s'$ is given by (7.3.7) and $s = s(\eta)$. 
Table 7.4.1. New link functions for the quantiles of some 1–parameter distributions.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>θ</th>
<th>Range of θ</th>
<th>Quantile function ξ_p</th>
<th>Link function [η(θ; p)]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>λ</td>
<td>(0, ∞)</td>
<td>−1/λ log(1 − p)</td>
<td>η(θ; p) = log (1 − p)^{-1/θ}</td>
</tr>
<tr>
<td>Benini</td>
<td>s</td>
<td>(0, ∞)</td>
<td>y_0 exp \left{ \sqrt{−\log(1 − p) θ} \right}</td>
<td>η(θ; p) = log y_0 + √{−\log(1 − p) θ}</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>b</td>
<td>(0, ∞)</td>
<td>b√−2 log(1 − p)</td>
<td>η(θ; p) = θ√−2 log(1 − p)</td>
</tr>
<tr>
<td>Gamma</td>
<td>s</td>
<td>(0, ∞)</td>
<td>No closed–form</td>
<td>η(θ; p) = log \gamma(p, shape = θ)</td>
</tr>
<tr>
<td>Maxwell†</td>
<td>a</td>
<td>(0, ∞)</td>
<td>√{2/α · qgamma(p, 1.5)}</td>
<td>η(θ; p) = √{2/α · qgamma(p, 1.5)}</td>
</tr>
<tr>
<td>Topp–Leone∥</td>
<td>s</td>
<td>(0,1)</td>
<td>1 − √{1 − p^{1/s}}</td>
<td>η(θ; p) = 1 − √{1 − p^{1/θ}} \text{ k.sup}</td>
</tr>
<tr>
<td>1–par Normal‡</td>
<td>σ</td>
<td>(0, ∞)</td>
<td>µ_0 ± √{2/θ^2 κ(p)}</td>
<td>η(θ; p) = µ_0 ± √{2/θ^2 κ(p)}</td>
</tr>
</tbody>
</table>

† qgamma() is the quantile function of the standard gamma distribution in R.
∥ k.sup = \sup_{0 < θ < 1} \left\{ 1 − √{1 − p^{1/θ}} \right\}.
‡ Here, κ(p) = erf^{−1}(2p − 1).

In this way the mean–link function of the logarithmic distribution logffMeanlink() is completely determined within the VGLM/VGAM context: the derivatives ∂^k g/∂(s)^k, for k = 0, 1, 2 are computed as in (7.3.4), while expressions (7.3.7) and (7.3.8), assisted by either newtonRaphson.basic() or VGAM::bisection.basic(), giving the required derivatives of the inverse, ∂^k g^{-1}/∂s^k. Further details on both implementations are provided in the VGAMextra and VGAM manuals. Finally, for the purpose of comparison, Figure 7.3.3 presents several 1–parameter probability link functions, including logffMeanlink(), with parameter range in (0,1).

7.4 On quantile modelling of several 1–parameter distributions

In a similar manner, methodology to directly model the quantile–function 1–parameter distributions outside the exponential family has been developed. Ordinarily, the associated VGLMs, say \mathcal{F}(η; y, x), handle a linear predictor η of the form

η = g(θ) = β^T x,

with g(·) a VGLM–link. In this approach the corresponding conditional quantile function, denoted here as Q_y (τ|x), is modelled and introduced in the VGLM through a
modified linear predictor, $\eta_\tau$, transformed as

$$
\eta_\tau = \mathcal{G}(Q_y(\tau|\mathbf{x})),
$$

(7.4.1)

with $\mathcal{G}$ a smooth and one–to–one function, and $\tau$ denoting the $\tau$th quantile. At present, this methodology has been implemented for several 1–parameter VGLMs where $\mathcal{G}$ is taken as either the log–link or the identity link, thus

$$
\eta_\tau = \log Q_y(\tau|\mathbf{x}) \quad \text{or} \quad \eta_\tau = Q_y(\tau|\mathbf{x}).
$$

(7.4.2)

The suitable choice varies according to the functional form of $Q_y(\tau|\mathbf{x})$. This method is called *conditional* (on $\mathbf{x}$) VGLM–quantile modelling, and essentially (7.4.1) is the only modification to the VGLM framework, giving place to a range of quantile–modelling options via generalized regression that may be thought of as a simpler, yet more elegant, alternative to well–known quantile regression techniques. Computationally, this framework for quantile modelling emerges in similar way as with the mean–links. That is, $\eta_\tau$ is embedded in the VGLM–framework via new quantile–link functions (available in VGAMextra 0.0-1) that handle the corresponding conditional quantile function (7.4.1) and are only compatible with VGLM family functions from VGAM.

Since its introduction in the late 1970s (Koenker and Basset, 1978), quantile regression has blossomed into a large subject with many extensions. A competing method is the so–called LMS–type quantile regression methods which are amenable to IRLS and resilient to the quantile crossing problem but incapable of handling multimodality (see, e.g., Yee (2004); Yee (2015), Section 15.2).

The quantile modelling methodology introduced with this work aims to accommodate the above as well and related issues by means of the VGLM/VGAM framework. The new links are shown in Table 7.4.1, while Table 7.4.2 gives the inverse and the function–name in VGAMextra. For fuller details, see the companion VGAMextra Manual. Similar to the mean–links from Section 7.3, the quantile–links proposed here are fully determined by its derivative–functions, as well as the inverse and its first two derivatives that have been derived upon the method given in Section 7.3.1.

### 7.4.1 An example with simulated data

To test out this methodology we will use simulated data ($n = 200$) sampled from a Maxwell distribution with rate parameter depending on a single covariate. To account for potential non–linear scatter in the dataset, additive models with say, cubic smoothing splines, appear to be a better choice over linear schemes such as VGLMs, and will be utilized to run the corresponding generalized fitting processes. A similar example is
Table 7.4.2. New link functions for the quantiles of 1–parameter distributions: Its inverse and names in VGAMextra. “Approximate” means that Newton–Raphson or bisection is used to approximate the inverse.

<table>
<thead>
<tr>
<th>Distribution</th>
<th>θ</th>
<th>Inverse $L[\theta(\eta; p)]$</th>
<th>Name in VGAMextra</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exponential</td>
<td>$\lambda$</td>
<td>$\theta(\eta; p) = \log (1 - p)^{-1/\eta}$</td>
<td>expQlink()</td>
</tr>
<tr>
<td>Benini</td>
<td>$s$</td>
<td>$\theta(\eta; p) = -\frac{\log(1 - p)}{(\eta - \text{loglink}(y_0))^2}$</td>
<td>benini1Qlink()</td>
</tr>
<tr>
<td>Rayleigh</td>
<td>$b$</td>
<td>$\theta(\eta; p) = \frac{\eta}{\sqrt{-2 \log(1 - p)}}$</td>
<td>rayleighQlink()</td>
</tr>
<tr>
<td>Gamma</td>
<td>$s$</td>
<td>Approximate</td>
<td>gamma1Qlink()</td>
</tr>
<tr>
<td>Maxwell</td>
<td>$a$</td>
<td>$\theta(\eta; p) = \frac{2}{\eta^2} \text{qgamma}(p, 1.5)$</td>
<td>maxwellQlink()</td>
</tr>
<tr>
<td>Topp–Leone</td>
<td>$s$</td>
<td>$\theta(\eta; p) = \frac{\log p}{\log [1 - (1 - \eta \cdot k.sup)^2]}$</td>
<td>toppleQlink()</td>
</tr>
<tr>
<td>1–par Normal</td>
<td>$\sigma$</td>
<td>$\theta(\eta; p) = \left[\frac{\eta - \mu_0}{\sqrt{2} \cdot \kappa(p)}\right]$</td>
<td>normal1sdQlink()</td>
</tr>
</tbody>
</table>

indeed addressed in Yee (2015, Section 15.3.3) to examine a methodology towards the crossing quantile problem via the so–called parallelism of the quantiles. Specifically, in this example we conduct the following, where the aim is to illustrate the performance of the methodology introduced with this work.

(1) First, we generate a random deviates from the Maxwell distribution.

(2) Second, we run conditional VGAM–quantile modelling using maxwellQlink() based on the VGAM family function VGAM::maxwell(), which estimates the Maxwell distribution by Fisher scoring. To assist on setting up the model matrix $X$, the function VGAMextra::Q.reg() is utilized.

(3) Then, we perform quantile regression using VGAM::alaplace1(), that estimates the 1–parameter asymmetric Laplace distribution by Fisher scoring. Here, the special argument tau will be employed.

(4) We finally give the plot of the artificial data with the estimated quantile–functions, $Q_y(\tau|x)$ (from (2)), and the estimated quantile curves (from (3)) superimposed.

We will consider the quantiles 25%, 50%, 75% for simplicity, yielding $\tau = (1/4, 1/2, 3/4)^T$, viz. the $t_{a}th$ regression quantile–vector.

Regarding (1), the data is generated upon VGAM::rmaxwell(), which gives random deviates from the Maxwell distribution with rate parameter–function defined as:

$$a = \exp \left\{2 - \frac{6 \sin \left(2x_{i2} - \frac{1}{2}\right)}{(x_{i2} + \frac{1}{2})^2}\right\},$$  \hspace{1cm} (7.4.3)
7.4. On quantile modelling of several 1–parameter distributions

Figure 7.4.1. Scatter plot of simulated Maxwell data (7.4.3) including (a) the fitted quantile functions from `fit.Qmodelling` (VGAMextra), and (b) the fitted quantile curves from `fit.Qregression` (VGAM). The quantile curves in both cases derive from vector smoothing spline fits. The points have been jittered slightly.

where $X_{i2} \overset{i.i.d}{\sim} \text{Unif}(0,1), i = 1,\ldots, 200$. The following code does this, where that dataset is saved in `maxdata`.

```r
# An artificial covariate, x2.
maxdata <- data.frame(x2 = sort(runif(n <- 200)))

# The 'rate' function.
mymu <- function(x) exp(2 - 6 * sin(2 * x - 0.2) / (x + 0.5)^2)

# Set up the data.
maxdata <- transform(maxdata, y = rmaxwell(n, rate = mymu(x2)))

# 25%, 50% and 75% quantiles are to be modelled.
my.tau <- c(0.25, 0.50, 0.75)

The following code chunk conducts (2) and (3) above. Note the use of additive models via `VGAM::vgam()` with smooth terms defined by `VGAM::s()` where the $x_2$ covariate is to be smoothed. To differentiate both fits, they are saved in `fit.Qmodelling` (from (2)) and `fit.Qregression` (from (3)).

```r
### Quantile modelling with 'maxwellQlink()'

```r
mydof <- 4  # Cubic splines effective degrees of freedom.
fit.Qmodelling <- vgam(Q.reg(y, pvector = my.tau) ~ s(x2, df = mydof),
                        family = maxwell(link = maxwellQlink(p = my.tau),
                                      type.fitted = "Qlink"),
                        data = maxdata)
```
Table 7.4.3. Data coverage from quantile–modelling using VGAMextra (QM–VGAMextra), and quantile–regression from VGAM (QR–VGAM), after fitting (7.4.3).

<table>
<thead>
<tr>
<th>Coverage</th>
<th>QM–VGAMextra</th>
<th>QR–VGAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>25% curve</td>
<td>23.5%</td>
<td>30%</td>
</tr>
<tr>
<td>50% curve</td>
<td>50%</td>
<td>54%</td>
</tr>
<tr>
<td>75% curve</td>
<td>74.5%</td>
<td>76.5%</td>
</tr>
</tbody>
</table>

### Quantile regression with alaplace2() from VGAM

```r
fit.Qregression <- vgam(y ~ s(x2, df = mydof), alaplace1(tau = my.tau, llocation = "loglink", parallel.locat = TRUE), data = maxdata, maxit = 100)
```

Note that the argument `p` of `maxwellQlink()` carries the quantiles of interest, which in this example is assigned to the three–entries numeric vector `my.tau = c(0.25, 0.50, 0.75)`. See above for `fit.Qmodelling`.

Finally, Figure 7.4.1 shows the simulated data slightly jittered, as well as the estimated quantile functions and the fitted quantile curves from `fit.Qmodelling` and `fit.Qregression`, obtained from vector smoothing spline fits. The curves appear to differ slightly from around the percentile 20%, however, the quantile–modelling alternative introduced with this work seems to suitably cope with the sharped departure at the bottom left (< 20% percentile), and further, with the quantile–crossing problem. Finally, the data coverage from each modelling framework is summarized in Table 7.4.3.

A few notes: (i) The argument `p` is available at all quantile–links in Table 7.4.2 (not only for `maxwellQlink()` in the above example) and can be assigned to any vector of percentiles of interest. (ii) Under the conditional VGAM–quantile modelling framework, the arguments to handle the parallelism assumption such as the arguments `parallel.locat` and `parallel.scale` in VGLMffs are not longer required. This is internally managed by the new quantile–links rather than being managed by VGLMffs.

### 7.5 Other VGLM family functions in VGAMextra

Besides the work on VGLTSMffs, VGAMextra comprises several other family functions for independent data, which were developed as early part of this work. Computationally, these new functions estimate the inverse–Weibull, the inverse–gamma, and the generalized beta of the second kind. VGAMextra also contains an implementation to estimate the multivariate normal distribution, however this VGLTSMff is restricted
Table 7.5.1. Other VGLMffs in VGAMextra.

<table>
<thead>
<tr>
<th>Name in VGAMextra</th>
<th>Linear predictors</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>invweibull2mr</td>
<td>$\eta_1 = \log s$, $\eta_2 = \log \alpha$.</td>
<td>Estimates the 2-parameter inverse Weibull distribution. The parameters are $s$, a scale parameter, and $\alpha$, a shape parameter.</td>
</tr>
<tr>
<td>invgamma2mr</td>
<td>$\eta_1 = \log \mu$, $\eta_2 = \log \alpha$.</td>
<td>Estimates the 2-parameter inverse gamma distribution. The parameters are $\mu$ (its mean) and $\alpha$, a shape parameter.</td>
</tr>
<tr>
<td>gen.betaIImr</td>
<td>$\eta_1 = \log \mu$, $\eta_2 = \log \alpha_1$, $\eta_2 = \log \alpha_2$, $\eta_2 = \log \alpha_4$.</td>
<td>Estimates the 4-parameter generalized beta distribution. The parameters are $s$ (scale) $\alpha_1, \alpha_2$, and $\alpha_3$ (shape parameters).</td>
</tr>
<tr>
<td>MVNcov</td>
<td>$\eta_{i,1} = \mu_i$, $\eta_{i,2} = \log \sigma_i^2$, $i = 1, \ldots, Q$.</td>
<td>Estimates the mean vector and the covariance matrix of the multivariate normal distribution ($Q$ responses). The correlation coefficients are not estimated.</td>
</tr>
</tbody>
</table>

(at its latest version) to estimating the covariance components only, ignoring the correlation parameters. In this context, VGAM::trinormal() which estimates the nine parameters of the trivariate normal distribution represents a better solution. A short description of such VGLMffs is given in Table 7.5.1.

The primary difference between the aforementioned VGLMffs and VGLTSMffs (Chapter 3) is the independence assumption on the observations which affects the computation of the EIMs, thus Fisher scoring. With independent data, the Fisher information for, say, $n$ independent observations is equivalent to $n$ times the Fisher information for a single observation. For VGLTSMs this is not true since an autocorrelation structure is assumed on the data, hence counterposing with the above statement.

### 7.6 Summary

A few topics relevant to VGLMs, but not VGLTSMs, were developed and have been presented in this chapter. In Section 7.2, VGLM–links were endowed with capabilities to handle $M = 2$ parameters. Meaningfully, with this method we have found the causes of the numerical instability observed when estimating by Fisher scoring the NB distribution with a canonical link, reported in Hilbe (2011, pages 210 and 309). This work signifies the initial steps towards somewhat broaden the VGLM modelling framework to incorporate linear predictors involving at least 2 parameters ($M > 2$). Another ex-
ample would be \texttt{VGAM::gamma2()}, the VGLMff that estimates the 2–parameter gamma
distribution, which currently handles $\eta_1(x) = \log \mu$ and $\eta_2(x) = \log \alpha$, where $\alpha$ is a
\textit{shape} parameter. In Section 7.3, the new set of 1–parameter mean–links represent
additional modelling alternatives for VGLM and VGAMs not available elsewhere. In
Section 7.4, the 1–parameter quantile–links have shown to produce analogous results as
with quantile regression techniques, but in a more elegant way of generalized “regres-
sion” and incorporating the “quantile modelling” component directly in the VGLM–log
likelihood.
Chapter 8

Future Work

8.1 Introduction

This work comprised three main contributions. Firstly, it introduced VGLTSMs as an extension of the GLM–type regression models for TS analysis by considering the VGLM log–likelihood with deterministic time–dependent information. Although confined to modelling and estimation, the intention was to show its advantages over a few preceding frameworks, e.g., those based on GLMs (Kedem and Fokianos, 2002). Secondly, it proposed methodology extending the VGLM–framework towards modelling and estimation with link functions of $M=2$ parameters, which supersedes GLMs and GAMLSS. The details of this approach were discussed in Chapter 7, where the NB–C2–2 model was used as an example. Thirdly, the resulting software VGAMextra, which extends the functionalities of VGAM with additional datasets and functions that address the aforementioned topics and others.

The VGLTSM–framework is a natural adaptation of the VGLM log–likelihood to handle TS data where the primary modifications were made over the score vector and the working weight matrices (shown in Chapter 2), the central elements for the Fisher scoring. Its implementation in VGAMextra is the class called VGLTSMffs, the sub–class of generic VGLM family functions that fully describe VGLTSMs, viz. VGLM–ARIMAX, VGLM–ARMAX–GARCH and VGLM–INGARCH, depicted in Chapter 2. Currently endowed with capabilities for modelling and estimation of TS, VGLTSMffs have demonstrated advantages over other modelling structures for TS analysis, as shown by several examples given in Chapters 4–6, such as ARIMAX.errors.ff() for dynamic regression (Section 3.4), or ARMA.studentt.ff(), a VGLTSMff for ARMA modelling where the errors are assumed to follow a Student–$t$ distribution (Section 3.4.3).
Further, we demonstrated the ability of VGLTSMs to handle multivariate time series, and a meaningful result was presented in Chapter 5 on cointegrated time series. There, we showed how bivariate systems of CTS are more conveniently and elegantly accommodated by VGLTSMs via the VGLTSMff $ECM.EngleGran()$ compared with other methods, e.g., Pfaff (2011). As well as $ECM.EngleGran()$ for CTS, VGLTSMs have been extended to manage VAR–type models, through $VARff()$ (Section 3.4.1).

A few other topics relevant to VGLMs were additionally developed, and the initial results were introduced in Chapter 7. These outcomes are further extensions of the VGLM–framework on mean–function and quantile–function modelling of several 1–parameter distributions beyond the GLM family, available in VGAMextra. Quantile–function modelling arises as an alternative to ordinary quantile regression techniques by simply incorporating the quantile information into the VGLM log–likelihood in the form of a VGLM–link. The improved performance of such compared to other classical methods for quantile regression that are included in VGAM was illustrated in Section 7.4.

8.2 Some directions for future work

The above provides fertile grounds for new directions that broaden the work further. In the following, some basic ideas are described for future work.

8.2.1 On forecasting with VGLTSM family functions

Presently, forecasting methods for VGLTSMs have been implemented at very initial stages, and including only intercept–only models from the sub–class VGLM–ARIMAX($u$, $0$, $v$), as covered in Section 2.4. The theory implemented here to estimate point forecasts is based on the minimum mean square error criteria, where the $l$th–step ahead is given by (2.4.3). Any VGLTSM fitted with family functions from the sub–class VGLM–ARIMAX($u$, $0$, $v$) is identified in the VGLTSM–framework by means of @object@family@vfamily, that should include a tag (identifier) like "ARMAvgltsmff". For such VGLTSMs, a generic forecast() functions and a generic show() has been implemented in VGAMextra.

Using artificial data, the following code gives an example of the output produced by forecast() for VGLM–ARIMAX($u$, $0$, $v$), which is compared with the forecasting output produced by forecast::forecast() (Hyndman and Khandakar, 2008). The function forecast for VGLTSMs is not included in VGAMextra 0.0-1 yet, and, for simplicity and illustration purposes, it has been renamed as forecast.VGAMextra().
First, we generate a sample \((n = 130)\) from an AR(2) model with coefficients \(\theta = (0.45, -0.21)^T\) and errors \(\varepsilon_t \overset{i.i.d.}{\sim} N(0, \sigma^2), \sigma = \sqrt{4.5}\), as follows. This series is stored in the data frame \(\text{tsdata}\) as \(\text{TS1}\).

```r
n <- 130
tsdata <- data.frame(x2 = runif(n)) # A single covariate.
ARcoeffs <- c(0.45, -0.21) # AR Coefficients
drift <- 1.3 # A drift (scaled mean)
sdAR <- sqrt(4.5) # The errors SD.
### Generate an AR(3), Gaussian noise.
tsdata <- transform(tsdata, TS1 = arima.sim(n = n, model = list(ar = ARcoeffs),
                       rand.gen = rnorm, mean = drift + tsdata$x2, sd = sdAR))
```

Next, we fit the best ARIMA model to the simulated series \(\text{TS1}\) using \texttt{auto.arima()}, which results in an AR(3) model. Based on this outcome, we thus fit an AR(3) using \texttt{VGAMextra::ARXff()}, as follows. The fitted models are saved as \texttt{fit.auto.arima} and \texttt{fit.ARXff}, respectively.

```r
> library("forecast")
> ### Fitting the best ARIMA model with \texttt{auto.arima()}
> ### The result is an AR(3).
> (fit.auto.arima <- with(tsdata, auto.arima(TS1)))

Series: TS1
ARIMA(3,0,0) with non-zero mean

Coefficients:
ar1  ar2  ar3  mean
 0.398 -0.203 -0.134  2.347
s.e.  0.087  0.093  0.089  0.209

sigma^2 estimated as 5.13: log likelihood=-288.84
AIC=587.67   AICc=588.16   BIC=602.01

> ### Fitting an intercept-only AR(3) with \texttt{VGAMextra::ARXff()}.
> fit.ARXff <- vglm(TS1 ~ 1, ARXff(order = 3, type.EIM = "exact"),
                       data = tsdata, trace = TRUE, lvar = FALSE)

VGLM linear loop 1: loglikelihood = -288.42692
VGLM linear loop 2: loglikelihood = -288.42692
VGLM linear loop 3: loglikelihood = -288.42692

Checks on stationarity / invertibility successfully performed.
No roots lying inside the unit circle.
Further details within the 'summary' output.

Now, we call \texttt{forecast.VGAMextra()} to estimate the point forecasts upon \texttt{fit.ARXff} for three periods ahead \((l = 3)\). This feature is managed by the argument \texttt{n.ahead} which should be a positive integer. For comparison purposes, the point forecasts returned by \texttt{forecast::forecast()} based on \texttt{fit.auto.arima} are also shown. The results are presented in the following output:

```r
> ###
> ### Point forecasts from \texttt{VGAMextra::forecast()}
> ###
> forecast.VGAMextra(fit.ARXff, n.ahead = 3)
```
The point forecasts are similar, however, these results can only be compared superficially as `forecast.VGAMextra` does not provide the estimated standard errors and consequently, it lacks the forecast confidence limits. In contrast, `forecast::forecast()` shows a superior performance by providing the point forecasts and confidence limits, and it is still equipped with further functionalities such as automated algorithms for forecasting. This is described in Hyndman and Khandakar (2008).

Nevertheless, `forecast.VGAMextra()` represents the basis and motivation to the development of forecasting methods for VGLTSMs. We will start with the class VGLM–ARIMAX, which is planned to be endowed with forecasting capabilities based on (a) typical benchmarks, such as naïve methods, and (b) stepwise–like algorithms for forecasting. Before this, however, VGLM–ARIMAX will need to be enhanced with capabilities for seasonal decomposition and estimation of seasonality. This work is primarily related to adapt the ‘big’ model matrix, $X_{VLM}$, and the VGLTSM log–likelihood accordingly. The prospective class of seasonal VGLM–ARIMAX models, is to be given a name such as VGLM–ARIMAX($p, d, q$)($P, D, Q$)$_k$, where $k$ is the seasonal frequency, and $d$ denotes the number of non–seasonal differences needed for stationarity. Note that procedures based on space state models are not considered at this stage.

### 8.2.2 On model selection

Model selection is another facility to be developed and implemented for VGLTSM family functions in `VGAMextra`. In the initial stage, only family functions from VGLM–ARIMAX($p, d, q$) with non–seasonal effects are considered, where a two step–procedure for model selection is planned to be implemented. The first stage will be devoted to
the selection of \( d \), where different methods have been proposed (see, e.g., Durbin and Koopman, 2001), however, for VGLM–ARIMAX\((p, d, q)\) unit roots tests, such as the KPSS test (Section 4.4.2), are to be considered. Next, in the second stage, the values of \( p, q \) will be selected by minimizing the information criteria AIC, BIC, and AICc, which is computed upon the VGLTSM log–likelihood using other functions, as depicted in Table 4.5.1. The seasonal VGLM–ARIMAX\((p, d, q)(P, D, Q)\) is contemplated at later stages.

The above methodology conforms with that implemented in `forecast::auto.arima()` to select, for a univariate time series, the order of the ARIMA model (seasonal or non–seasonal) that performs the best using unit root tests and the information criteria AIC, among others. However, for non–seasonal ARIMAs (handled by the logical argument `seasonal`), `auto.arima()` is restricted to errors \( \{\varepsilon_t\} \) that resemble white noise, therefore \( \sigma^2_{\varepsilon_t} \) must be intercept–only, as well as other conditions on the parameters, such as causality and invertibility (Section 4.4.3). By default, VGLM–ARIMAX incorporates \( \sigma^2_{\varepsilon_t} \) in a linear predictor dependent on explanatory variables (if this is intercept–only, then it reduces to the case scenario operated by `forecast`), and then it is embedded in the VGLM log–likelihood. Prospective work to be developed in this line relates to exploring how the two–step criteria proposed for models selection with VGLM–ARIMAX needs to be adapted in the presence of non–stationary or non–causality conditions, and particularly, how the expressions to estimate accuracy measures, such as the MSE, need to be modified.

Moreover, \( \sigma^2_{\varepsilon_t} \) is a common parameter shared by VGLM–ARIMAX (eq. (2.2.1)) and VGLM–ARMAX–GARCH (eq. (2.2.2)), which is handled by linear predictors that are structurally tied by the component \( \beta^{T}_{2,K}x_{t(1)} \), including other terms on deterministic past information (see Tables 2.2.1 and 2.2.3). Consequently, automatic model selection for VGLM–ARMAX–GARCH may be nothing but a natural extension of model selection techniques pertaining to VGLM–ARIMAX. This is another topic that warrants further development, but is planned to be addressed at more advanced stages.

### 8.2.3 On detecting interventions with `VGLM.INGARCHff()`

The class VGLM–INGARCH (Section 2.2.3) is a general regression model for the analysis of TSCs that allows intervention analysis with the ability to explore joint effects (see equation (2.2.16)). Its implementation in VGAMextra is the VGLTSMff `VGLM.INGARCHff()` (see Table 3.3.4), which makes VGLM–INGARCH operational upon distributional assumptions on the response beyond the exponential family, as shown in Table 2.2.5. However, a significant demerit of VGLM–INGARCH to date is that it lacks hypothesis testing methodology to automatically investigate the effect of
a set of arbitrary interventions on the series occurring, say, at \( k \) unknown time points. Formally, \( H_0 : \omega_1 = \cdots = \omega_k = 0 \), with \( \omega_j \) as in (2.2.19).

For the INGARCH class of models, Fokianos and Fried (2012), Liboschik et al. (2016), and Liboschik et al. (2017), have proposed methodology based on asymptotic iterative–detection algorithms that are built on multiple testing using Bonferroni adjustments. Here, the \( p \)-values are approximated via bootstrapping. Thus, at each iteration, the lowest \( p \)-value identifies a potential intervention, which is disregarded from the time series in the next step, until no further interventions are detected. The procedure from Liboschik et al. (2016, 2017) provides a benchmark for VGLM–INGARCH to be endowed with such capabilities. Future work along this line relates to investigate whether VGLTSMs are amenable to multiple testing using Bonferroni adjustments and the conditions that need to be met for its success.

### 8.2.4 Some comments on VGLTSMs and on VGAMextra

Several useful features for TS models required by VGAMextra prompted specific developmental and generic extensions from VGAM 1.0-0 onwards, which coincides with the appearance of Yee (2015). They include the following (carried out by its maintainer):

- **The zero argument**, that allows partial character matching using `grep()`.
- **VGAM 1.0-0**, `zero` was restricted to positive integers to specify the intercept–only linear predictors, but under the assumption that the number of parameters, \( M \), in the VGLM was fixed. For instance, `zero = c(2,4)` indicates that \( \eta_{t,2} = \beta_{(2),1} \) and \( \eta_{t,4} = \beta_{(4),1} \) were intercept–only.

From **VGAM 1.0-0** this can be overridden by characters where, e.g., `zero = "coeff"` constrains the \( \eta_j \) corresponding to parameters `zero = "ARcoeff"` and `zero = "MAcoeff"` to be intercept–only. This is much more convenient for TS analysis with VGAMextra as it conveys greater readability to many VGLTSM family functions such as `ARMAXff()` and `ARMAX.GARCHff()`, where the number of parameters \( M \) cannot be determined a priori. That is, for instance, for \( \eta_t = \left( \mu^*, \sigma_{\varepsilon_t}^2, \varphi_{t-1}, \theta_1, \ldots, \theta_S^* \right)^T \) in an AR model, \( S^* \) cannot be determined until the required order is entered and processed by `vglm()`.

- **Additionally**, `summary` methods for VGLMs (Section 18.3 in Yee, 2015) needed to be enhanced to allow TS output at the bottom of the default VGLM `summary`, as shown in Section 3.5, prompting, e.g., the adjusted standard errors based on the VGLTSM covariance (2.3.14). This feature was made available by further S4–OOP programming. See also Appendix A.1 for an example that shows the current summary output for VGLTSMs.
Some other topics beyond Sections 8.2.1–8.2.3 may be subject of further investigation and implementations concerning VGLMs and VGLTSMs. For instance, the extension of the VGLTSM-framework to sources of noise other than normal, such as Cauchy errors. At present, \texttt{ARMA.studentt.ff()} represents a first approach in this direction. Another topic in this line may be the extension to VGAMs, the non-parametric version of VGLMs, to handle time series. The result would be something similar to the non-parametric version of VGLTSMs for TS analysis, e.g., vector generalized additive time series models or VGATSMs. But this work is deferred to more advanced stages.
Appendices
Appendix A

A.1 Summary of model `fit.PM10` using VGAMextra

The following summary of `fit.PM10`, from section 6.2, is an example to illustrate the flexibility of VGLTSM on handling parameter constraints. For this, the corresponding `summary()` method implemented for VGLTSMs is called. It should be noted that, by default, `summary()` returns two sets of estimated standard errors that conform with (a) the VGLM covariance matrix (1.3.15), and with (b) the adjusted VGLTSM covariance matrix (2.3.14), as illustrated in section 3.5.1.

```r
> summary(fit.PM10)

Call:
vglm(formula = PM10 ~ O3_lag1 + O3_lag2 + temp_lag1 + temp_lag2 +
      temp_lag3 + HR_lag1 + HR_lag2, family = ARMAXff(order = c(1,
      2), var.arg = FALSE, nodrift = TRUE, type.EIM = "exact",
      noChecks = TRUE, zero = c("ARcoeff", "MAcoeff")), data = ap.mx2)

Pearson residuals:
                Min     1Q  Median     3Q    Max
loglink(noiseSD) -0.707 -0.6145 -3.75e-01 0.1487 10.384
ARcoeff1  -3.811 -0.5975 1.16e-01 0.6989 2.915
MAcoeff1    -1.283 -0.0824 6.00e-03 0.0979 0.927
MAcoeff2    -1.075 -0.0870 -7.06e-08 0.0914 29.510

Coefficients:
                        Estimate  Std. Error   z value     Pr(>|z|)
(Intercept):1  4.37729      0.25717  17.02 < 2e-16 ***
(Intercept):2  0.99006      0.00797 124.24 < 2e-16 ***
(Intercept):3 -0.29296      0.04262  -6.87 6.2e-12 ***
(Intercept):4 -0.31180      0.04056  -7.69 1.5e-14 ***
O3_lag1       -0.00269      0.00101  -2.68  0.0074 **
O3_lag2        0.00211      0.00100   2.11  0.0351 *
temp_lag1     -0.04035      0.02636  -1.53   0.1259
temp_lag2     -0.05711      0.03400  -1.68   0.0930 .
temp_lag3      0.03710      0.02294   1.62   0.1058
HR_lag1       -0.01381      0.00509  -2.71   0.0067 **
HR_lag2       -0.00257      0.00499  -0.52   0.6056
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

191
Number of linear predictors: 4

Names of linear predictors: loglink(noiseSD), ARcoeff1, MAcoeff1, MAcoeff2

Log-likelihood: -2097 on 2177 degrees of freedom

Number of iterations: 12

** Standard errors based on the asymptotic distribution of the MLE estimates:

<table>
<thead>
<tr>
<th></th>
<th>ARcoeff1</th>
<th>MAcoeff1</th>
<th>MAcoeff2</th>
</tr>
</thead>
<tbody>
<tr>
<td>s.e.</td>
<td>0.990</td>
<td>-0.293</td>
<td>-0.312</td>
</tr>
</tbody>
</table>

Estimated linear predictor of sigma^2 (SD errors):

<table>
<thead>
<tr>
<th></th>
<th>(Intercept)</th>
<th>O3_lag1</th>
<th>O3_lag2</th>
<th>temp_lag1</th>
<th>temp_lag2</th>
<th>temp_lag3</th>
<th>HR_lag1</th>
<th>HR_lag2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated value</td>
<td>4.377289</td>
<td>-0.002690</td>
<td>0.002109</td>
<td>-0.040345</td>
<td>-0.057111</td>
<td>0.037096</td>
<td>-0.013806</td>
<td>-0.002575</td>
</tr>
<tr>
<td>Loglikelihood</td>
<td>-2096.83</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIC</td>
<td>4201.65</td>
<td>AICC 4201.73</td>
<td>BIC 4218.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** Summary of checks on stationarity / invertibility:

Polynomial roots of the AR component computed from the estimated coefficients: (Examining stationarity/invertibility)

Model1
Root1 1.0101

Polynomial roots of the MA component computed from the estimated coefficients: (Examining stationarity/invertibility)

Model1
Root1 1.38129
Root2 2.32039
A.2 Summary of models fit.gamlss.arima and fit.gamlss.armax

We present here some additional outputs from the estimated fit.gamlss.arima and fit.gamlss.armax, from Section 6.2, fitted with GAMLSS::gamlss. We firstly show the summary of fit.gamlss.arima.

```r
> ## Results of fit.gamlss.arima
> summary(fit.gamlss.arima)

Family: c("NO", "Normal")
Call: gamlss(formula = PM10 ~ PM10_lag1 + error_lag1 + error_lag2,
  sigma.formula = ~O3_lag1 + O3_lag2 + temp_lag1 +
  temp_lag2 + temp_lag3 + HR_lag1 + HR_lag2,
  family = NO(mu.link = "identity", sigma.link = "log"),
  data = ap.mx3, trace = FALSE)
Fitting method: RS()

Mu link function: identity
Mu Coefficients:
  Estimate  Std. Error t value Pr(>|t|)
(Intercept)  4.84432 1.65958  2.919 0.00366 **
PM10_lag1    0.90612 0.03056 29.654 <2e-16 ***
error_lag1   -0.19384 0.04896 -3.959 8.53e-05 ***
error_lag2   -0.26150 0.04595 -5.690 2.09e-08 ***

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Sigma link function: log
Sigma Coefficients:
  Estimate  Std. Error t value Pr(>|t|)
(Intercept)  4.36264  0.25637 17.017 <2e-16 ***
O3_lag1   -0.00277  0.00102  2.706 0.00702 **
O3_lag2   -0.00207  0.00102  2.030 0.04282 *
O3_lag3   -0.03058  0.02590  1.181 0.23822
temp_lag1 -0.030583 0.02590 -1.181 0.23822
temp_lag2 -0.06431  0.03934 -1.634 0.10278
temp_lag3  0.035745  0.02959  1.208 0.22765
HR_lag1   -0.01117  0.00508  2.197 0.02843 *
HR_lag2   -0.00504  0.00476 -1.060 0.28972

---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

No. of observations in the fit: 547
Degrees of Freedom for the fit: 12
Residual Deg. of Freedom: 535
at cycle: 4
```
Now we give the summary of `fit.gamlss.armax`.

```r
> ### Results of fit.gamlss.armax
> summary(fit.gamlss.armax)
```

**Global Deviance:** 4194.639
**AIC:** 4218.639
**SBC:** 4270.293

******************************************************************

Now we give the summary of `fit.gamlss.armax`.

```r
> ### Results of fit.gamlss.armax
> summary(fit.gamlss.armax)
```

**Global Deviance:** 4197.052
**AIC:** 4221.052
**SBC:** 4272.706

******************************************************************
A.3 Summary of model (6.3.2) estimated with both, VGLM.INGARCH() and tscount::tsglm()

Here is the summary of model (6.3.2) estimated with tscount::tsglm(), and with VGLM.INGARCH() with and without interventions. The corresponding summary() function is used. The first output is the summary of (6.3.2) fitted with tsglm().

```r
> ### Summary from 'campytsglm.int' using 'tsglm()'.
> summary(campytsglm.int)

Call:
tsglm(ts = campy, model = list(past_obs = 1, past_mean = 13),
      xreg = interventions, link = "log", distr = "nbinom")

Coefficients:
                  Estimate Std.Error CI(lower) CI(upper)
(Intercept)    0.79320   0.20450   0.3920   1.1940
beta_1         0.38000   0.07470   0.2340   0.5260
alpha_13       0.22770   0.10190   0.0280   0.4270
interv_1       0.38710   0.09130   0.2080   0.5660
interv_2       1.15580   0.21960   0.7250   1.5860
interv_3       0.22720   0.28570  -0.3330   0.7870
interv_4      -1.01210   0.47810  -1.9490  -0.0750
sigmasq        0.02850          NA          NA          NA

Standard errors and confidence intervals (level = 95 %) obtained
by normal approximation.

Link function: log
Distribution family: nbinom (with overdispersion coefficient 'sigmasq')
Number of coefficients: 8
Log-likelihood: -378.4155
AIC: 772.8311
BIC: 796.3642
QIC: 786.5913
```

The following is the summary of the estimated model (6.3.2) without interventions fitted with VGLM.INGARCH().

```r
> summary(campy_vglm.NOint)

Call:
vglm(formula = y ~ 1, family = VGLM.INGARCHff(dist.type = "negbinom",
Order = c(1, 0), lagged.fixed.means = 13, interventions = list(tau = c(84,
100, 113, 133), delta = c(0.99, 0, 0.5, 0), No.Inter = TRUE),
link = "loge", init.p.ARMA = 24), data = campy_tscount, trace = FALSE)

Pearson residuals:
                  Min  1Q Median  3Q    Max
loglink(mu1) -2.271 -0.6530 -0.05495 0.7600 2.3272
size1        -3.938 -0.2253  0.32314 0.6544 0.7379
```
Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) |
|----------|------------|---------|----------|
| (Intercept):1 | 1.829636 | 0.061209 | 29.891 < 2e-16 *** |
| (Intercept):2 | 39.718156 | 20.928742 | 1.898 0.0577 . |
| Ylag1 | 0.022391 | 0.003892 | 5.754 8.73e-09 *** |
| lambLag13 | 0.011737 | 0.004632 | 2.534 0.0113 * |
| Interv.1 | 0.450185 | 0.090926 | 4.951 7.38e-07 *** |
| Interv.2 | 1.224708 | 0.214279 | 5.715 1.09e-08 *** |
| Interv.3 | 0.450164 | 0.244543 | 1.841 0.0656 . |
| Interv.4 | -0.952998 | 0.475947 | -2.002 0.0453 * |

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors: 2

Names of linear predictors: loglink(mu1), size1

Log-likelihood: -377.6405 on 272 degrees of freedom

Number of iterations: 10

The summary of the estimated model (6.3.2) with interventions and fitted with `VGLM.INGARCH()` is the following:

```r
> summary(campy_vglm.int)
```

Call:

```r
vglm(formula = y ~ 1, family = VGLM.INGARCHff(dist.type = "negbinomial",
    Order = c(1, 0), lagged.fixed.means = 13, interventions = list(tau = c(84,
        100, 113, 133), delta = c(0.99, 0, 0.5, 0), No.Inter = FALSE),
    link = "loge", init.p.ARMA = 24), data = campy_tscount, trace = FALSE)
```

Pearson residuals:

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>loglink(mu1)</td>
<td>-2.386</td>
<td>-0.7224</td>
<td>-0.04839</td>
<td>0.8020</td>
<td>2.3675</td>
</tr>
<tr>
<td>size1</td>
<td>-4.179</td>
<td>-0.3477</td>
<td>0.27025</td>
<td>0.6454</td>
<td>0.7366</td>
</tr>
</tbody>
</table>

Coefficients:

| Estimate | Std. Error | z value | Pr(>|z|) |
|----------|------------|---------|----------|
| (Intercept):1 | 1.776e+00 | 6.253e-02 | 28.403 < 2e-16 *** |
| (Intercept):2 | 6.195e+01 | 4.556e+01 | 1.360 0.173952 |
| Ylag1 | 2.300e-02 | 3.589e-03 | 6.408 1.48e-10 *** |
| lambLag13 | 1.792e-02 | 4.978e-03 | 3.601 0.000317 *** |
| Interv.2 | 1.220e+00 | 1.914e-01 | 6.375 1.83e-10 *** |
| Interv.3 | -1.072e+02 | 4.002e+01 | -2.678 0.007407 ** |
| Interv.4 | 9.820e-01 | 4.663e-01 | -2.106 0.035218 * |
| I1I3    | 1.442e+02 | 5.363e+01 | 2.689 0.007161 ** |

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors: 2

Names of linear predictors: loglink(mu1), size1

Log-likelihood: -374.2031 on 271 degrees of freedom

Number of iterations: 10
A.4 Complementary material for model (6.3.4)

This Appendix gives complementary estimation details of model (6.3.4) where the extended–GAM and VGLM–INGARCH are contrasted when analyzing the cumulative effects of air pollution on hospital admissions due to respiratory diseases in Hong Kong. The outcomes provided here are:

A) Figure A.4.1m that shows the estimated admissions obtained from the extended–GAM model proposed by Xia and Tong (2006b), and

B) The summary of model (6.3.4) fitted with `VGLM.INGARCHff()`, saved as `fit22_nb`.

![Figure A.4.1](image)

Figure A.4.1. Fitted values obtained with the extended–GAM model.

We start with Figure A.4.1, where blue is the notified number of admissions and the red are the fitted values obtained with the extended–GAM. The cyan line are the fitted admissions with GAM only. Note that the fitted values from `VGLM.INGARCHff()` are not included here. These are shown in Figure 6.3.7.

Now we give the summary of model (6.3.4) fitted with `VGLM.INGARCHff()` where hospital admissions is assumed as distributed as negative binomial:

```r
> ### Model fit22_nb estimated with 'vglm()' using 'VGLM.INGARCHff()'
> summary(fit22_nb)

Call:
vglm(formula = resp ~ fri + sat + o3 + rsp + no2lag1 + so2lag1 +
    rsplag1 + o3lag1 + templag1 + humlag1 + no2lag2 + so2lag2 +
    rsplag2 + o3lag2 + templag2 + humlag2, family = VGLM.INGARCHff(dist.type = "negbinomial",
    Order = c(1, 1), link = "loglink", interventions = list(),
    lagged.fixed.means = c(7, 14, 30, 90, 180, 365)), data = HKdata.2,
    trace = FALSE)
```
Pearson residuals:

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>loglink(mu1)</td>
<td>-2.562</td>
<td>-0.7625</td>
<td>0.04331</td>
<td>0.6953</td>
<td>3.7573</td>
</tr>
<tr>
<td>size1</td>
<td>-7.977</td>
<td>-0.2230</td>
<td>0.32129</td>
<td>0.6356</td>
<td>0.7093</td>
</tr>
</tbody>
</table>

Coefficients:

|               | Estimate | Std. Error | z value | Pr(>|z|) |
|---------------|----------|------------|---------|---------|
| (Intercept):1 | 4.614e+00 | 4.309e-02  | 107.073 | < 2e-16 *** |
| (Intercept):2 | 1.719e+02 | 1.364e+01  | 12.600  | < 2e-16 *** |
| fri1          | -5.609e-02 | 9.628e-03  | -5.826  | 5.68e-09 *** |
| sat1          | -2.715e-02 | 9.671e-03  | -2.807  | 0.005001 ** |
| o3            | -3.366e-04 | 1.856e-04  | -1.814  | 0.069689 . |
| rsp           | 5.556e+04 | 1.979e+04  | 2.807   | 0.004994 ** |
| no2lag1       | -2.152e-04 | 4.133e-04  | -0.521  | 0.602565 |
| so2lag1       | -6.555e-04 | 4.617e-04  | -1.423  | 0.033732 * |
| o3lag1        | 6.862e-03 | 2.373e-03  | 2.892   | 0.003830 ** |
| templag1      | 7.998e-04 | 5.568e-04  | 1.435   | 0.151344 |
| humlag1       | 2.143e-04 | 3.984e-04  | 0.538   | 0.590657 |
| so2lag2       | 4.715e-04 | 4.517e-04  | 1.044   | 0.296597 |
| rsplag2       | -2.187e-04 | 2.637e-04  | -0.830  | 0.406782 |
| o3lag2        | -1.005e-02 | 2.386e-03  | -4.211  | 2.54e-05 *** |
| templag2      | -1.897e-03 | 3.543e-04  | -3.552  | 0.000383 *** |
| humlag2       | 1.624e-03 | 1.901e-04  | 8.543   | < 2e-16 *** |
| lamblag1      | 1.157e-03 | 2.416e-04  | 4.789   | 1.68e-06 *** |
| lamblag7      | 1.288e-03 | 1.784e-04  | 7.220   | 5.19e-13 *** |
| lamblag14     | 8.021e-05 | 1.545e-04  | 0.519   | 0.603679 |
| lamblag30     | 4.101e-05 | 1.051e-04  | 0.390   | 0.696509 |
| lamblag90     | 1.290e-04 | 8.746e-05  | 1.475   | 0.140118 |
| lamblag180    | -1.216e-04 | 6.216e-05  | -1.956  | 0.050436 . |
| lamblag365    | 1.132e-04 | 5.649e-05  | 2.004   | 0.045120 * |

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of linear predictors: 2

Names of linear predictors: loglink(mu1), size1

Log-likelihood: -4857.85 on 2154 degrees of freedom

Number of iterations: 8

For responses distributed as Poisson, its summary is

> summary(fit22)

Call:

vglm(formula = resp ~ fri + sat + o3 + rsp + no2lag1 + so2lag1 +
     rsplag1 + o3lag1 + templag1 + humlag1 + no2lag2 + so2lag2 +
     rsplag2 + o3lag2 + templag2 + humlag2, family = VGLM.INGARCHff(dist.type = "poisson",
     Order = c(1, 1), link = "loglink", interventions = list(),
     lagged.fixed.means = c(7, 15, 30, 90, 180, 366)), data = HKdata.2,
     trace = FALSE)

Pearson residuals:
### A.4. Complementary material for model (6.3.4)

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>1Q</th>
<th>Median</th>
<th>3Q</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>loglink(lambda)</td>
<td>-4.053</td>
<td>-1.114</td>
<td>0.08725</td>
<td>1.003</td>
<td>5.373</td>
</tr>
</tbody>
</table>

Coefficients:

| Term            | Estimate | Std. Error | z value | Pr(>|z|) |
|-----------------|----------|------------|---------|---------|
| (Intercept)     | 4.627e+00 | 2.959e-02  | 156.388 | < 2e-16 *** |
| fri1            | -5.429e-02 | 6.571e-03  | -8.261  | < 2e-16 *** |
| sat1            | -2.942e-02 | 6.598e-03  | -4.459  | 8.24e-06 *** |
| o3              | -3.147e-04 | 1.243e-04  | -2.532  | 0.011344 * |
| rsp             | 5.682e-04  | 1.328e-04  | 4.279   | 1.88e-05 *** |
| no2lag1         | -2.798e-04 | 2.784e-04  | -1.005  | 0.316942   |
| so2lag1         | -6.675e-04 | 3.093e-04  | -2.158  | 0.030924 * |
| rsplag1         | -5.614e-04 | 1.857e-04  | -3.024  | 0.002493 ** |
| o3lag1          | 8.859e-04  | 1.447e-04  | 6.120   | 9.34e-10 *** |
| templag1        | 7.186e-03  | 1.604e-03  | 4.479   | 7.51e-06 *** |
| humlag1         | 7.711e-04  | 3.801e-04  | 2.029   | 0.042483 * |
| no2lag2         | 1.260e-04  | 2.696e-04  | 0.467   | 0.640185   |
| so2lag2         | 5.761e-04  | 3.048e-04  | 1.890   | 0.068731   |
| rsplag2         | -2.258e-04 | 1.779e-04  | -1.270  | 0.204248   |
| o3lag2          | -2.562e-04 | 1.384e-04  | -1.851  | 0.064110   |
| templag2        | -1.018e-02 | 1.610e-03  | -6.325  | 2.54e-10 *** |
| humlag2         | -1.753e-03 | 3.644e-04  | -4.812  | 1.49e-06 *** |
| Ylag1           | 1.605e-03  | 1.279e-04  | 12.552  | < 2e-16 *** |
| lambLag1        | 1.193e-03  | 1.615e-04  | 7.387   | 1.51e-13 *** |
| lambLag7        | 1.521e-03  | 1.091e-04  | 13.945  | < 2e-16 *** |
| lambLag15       | -4.275e-04 | 9.744e-05  | -4.387  | 1.15e-05 *** |
| lambLag30       | 1.893e-04  | 7.341e-05  | 2.578   | 0.009929 ** |
| lambLag90       | 1.024e-04  | 5.917e-05  | 1.731   | 0.083422   |
| lambLag180      | -1.414e-04 | 4.266e-05  | -3.314  | 0.000919 *** |
| lambLag366      | 1.717e-04  | 3.808e-05  | 4.508   | 6.54e-06 *** |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Number of linear predictors: 1

Name of linear predictor: loglink(lambda)

Log-likelihood: -5064.992 on 1065 degrees of freedom

Number of iterations: 4
Bibliography


