ISSN 1471-0498

DEPARTMENT OF ECONOMICS

DISCUSSION PAPER SERIES

General-to-Specific (GETS) Modelling And Indicator Saturation With The R Package Gets

Felix Pretis, James Reade, and Genaro Sucarrat

Number 794
April 2016

Manor Road Building, Oxford OX1 3UQ

General-to-Specific (GETS) Modelling and Indicator Saturation with the R Package gets (version 0.7)

Felix Pretis
University of Oxford

James Reade
University of Reading

Genaro Sucarrat
BI Norwegian Business School

(11th April 2016)

Abstract

This paper provides an overview of the R package **gets**, which contains facilities for General-to-Specific (GETS) modelling of the mean and variance of a regression, and Indicator Saturation (IS) methods for the detection and modelling of structural breaks and outliers. The mean can be specified as an autoregressive model with covariates (an 'AR-X' model), and the variance can be specified as an autoregressive log-variance model with covariates (a 'log-ARCH-X' model). The covariates in the two specifications need not be the same, and the classical regression model is obtained as a special case when there is no dynamics, and when there are no covariates in the variance equation. The four main functions of the package are `arx`, `getsm`, `getsv` and `isat`. The first function estimates an AR-X model with log-ARCH-X errors. The second function undertakes GETS model selection of the mean specification of an `arx` object. The third function undertakes GETS model selection of the log-variance specification of an `arx` object. The fourth function undertakes GETS model selection of an indicator saturated mean specification allowing for the detection of structural breaks and outliers. Examples of how LaTeXcode of the estimation output can be generated is given, and the usage of two convenience functions for export of results to **EViews** and **STATA** are illustrated.

Keywords: General-to-Specific, model selection, regression of the mean, regression of the log-variance, time series, AR-X, log-ARCH-X, indicator saturation, R.

Contents:

Introduction 2

GETS-modelling 3

Setting time-series attributes 4

Yearly, quarterly and monthly data 4

Weekly, daily and higher frequencies 5

The AR-X model with log-ARCH-X errors 5

Simulation . . . 7
`arx`: Estimation . . . 7
Extraction functions . . . 9
Example: A model of quarterly inflation with time-varying conditional variance . . . 9
Example: A rich model of daily SP500 volatility . . . 12

GETS modelling **14**
getsm: Modelling the mean . . . 14
getsv: Modelling the log-variance . . . 18
Extraction functions . . . 18
Example: A parsimonious model of quarterly inflation . . . 18
Example: A parsimonious model of daily SP500 volatility . . . 21

Indicator saturation **22**
Example: Structural breaks in UK SO_2 emissions growth . . . 23
Testing in isat . . . 24

Generating LaTeX code **27**

Exporting results to EViews and Stata **28**

References **31**

1. Introduction

General-to-Specific (GETS) modelling combines well-known ingredients: Backwards elimination, single and multiple hypothesis testing, goodness-of-fit measures and diagnostics tests. The way these ingredients are combined enables rival theories and models to be tested against each other, ultimately resulting in a parsimonious, statistically valid model that explains the characteristics of the data being investigated. The methodology thus provides a systematic and coherent approach to cumulative research and scientific progress.

The origins of GETS modelling can be traced back to Denis Sargan and the London School of Economics (LSE) during the 1960s, see Hendry (2003) and Mizon (1995). However, it was not until the 1980s and 1990s that the methodology gained widespread acceptance and usage in economics, with Sir David F. Hendry in particular being a main proponent, see the two-volume article collection by Campos, Hendry, and Ericsson (2005) for a comprehensive overview of the GETS methodology. A major software-contribution to the GETS literature was made in 1999, when Hoover and Perez (1999) re-visited Lovell's (1983) data-mining experiment. Hoover and Perez (1999) showed that automated multi-path GETS modelling substantially improved upon the then (in economics) popular model selection strategies. In the study of Hoover and Perez (1999), purpose-specific but limited MATLAB code was used in the simulations.[1] Subsequently, further improvements were achieved in the commercial

[1] The data and MATLAB code used is available from here: http://www.feweb.vu.nl/econometriclinks/journal/volume2/HooverKD_PerezSJ/data_and_code/.

software packages **PcGets**, see Hendry and Krolzig (2001)), and in its successor **Autometrics**, see Doornik and Hendry (2007). In particular, indicator–saturation methods for the detection of outliers and structural breaks proposed by Hendry, Johansen, and Santos (2007) were added to **Autometrics** in 2008, see Doornik (2009). Another milestone was reached in 2011, when the R package **AutoSEARCH** was published on the CRAN. The package, whose code was developed in relation with the research project Sucarrat and Escribano (2012), offered automated GETS modelling of conditional variance specifications within the log-ARCH-X class of models. The R package **gets**, available from the CRAN since October 2014, is the successor of **AutoSEARCH**. The **gets** package is the only statistical software that offers GETS modelling of the conditional variance of a regression, in addition to GETS modelling of the mean of a regression, and Indicator Saturation (IS) methods for the detection of outliers and breaks in the mean of a regression using impulses (IIS), step- (SIS see Castle, Doornik, Hendry, and Pretis 2015) as well as trend-indicators.

This paper provides an overview of the **gets** package. The model class under consideration is the Autoregressive (AR) model with exponential Autoregressive Conditional Heteroscedastic (ARCH) variance, possibly with additional covariates in the mean or variance equations, or in both. In short, the AR-X model with a log-ARCH-X error term, where the "X" refers to the covariates (the covariates need not be the same in the mean and variance specifications). The next section, Section 2, provides an overview of GETS modelling. Section 3 shows how data can be given time-series attributes, which is useful for optimal estimation output and graphing. Section 4 contains an overview of the AR-X model with log-ARCH-X errors, explains how it can be simulated, and illustrates how it can be estimated with the `arx` function. Section 5 illustrates how GETS modelling can be undertaken with the the `getsm` and `getsv` functions. The first undertakes GETS modelling of the mean specification, whereas the second undertakes GETS modelling of the log-variance specification. Section 6 introduces the `isat` function for indicator saturation methods. Section 7 shows how estimation output can readily be converted into LaTeXcode. Section 8 illustrates how two convenience functions, `eviews` and `stata`, facilitates GETS modelling by users whose primary work environment is either **EViews** or **STATA**, i.e. the two most popular commercial softwares in econometrics.

2. GETS-modelling

It is convenient to provide an overview of GETS modelling in terms of the linear regression model

$$y_t = \beta_1 x_{1t} + \cdots + \beta_k x_{kt} + u_t, \qquad t = 1, 2, \ldots, n, \tag{1}$$

where u_t is a zero mean error term. GETS modelling assumes there exists at least one "local" Data Generating Process (LDGP) nested in (1). By philosophical assumption *the* DGP is not contained in the simple model above, see Sucarrat (2010) and Hendry and Doornik (2014, Sections 6.2-6.3). The qualifier "local" thus means it is assumed that there exists a specification within (1) that is a statistically valid representation of *the* DGP. Henceforth, for notational and theoretical convenience, we will assume there exists only a single LDGP, but this is not a necessary condition.

A variable x_{jt}, $j \in \{1, \ldots, k\}$, is said to be relevant if $\beta_j \neq 0$ and irrelevant if $\beta_j = 0$. Let $k_{rel} \geq 0$ and $k_{irr} \geq 0$ denote the number of relevant and irrelevant variables, respectively, such that $k_{rel} + k_{irr} = k$. Of course, k_{rel} and k_{irr} are unknown to the investigator. GETS

modelling aims at finding a specification that contains as many relevant relevant variables as possible, and a proportion of irrelevant variables that corresponds to the significance level α chosen by the investigator. Put differently, if $\widehat{k}_{rel}$ and $\widehat{k}_{irr}$ are the retained number of relevant and irrelevant variables, respectively, then GETS modelling aims at satisfying

$$E(\widehat{k}_{rel}/k_{rel}) \to 1 \quad \text{and} \quad E(\widehat{k}_{irr}/k_{irr}) \to \alpha \quad \text{as} \quad n \to \infty, \tag{2}$$

when $k_{rel}, k_{irr} > 0$. If either $k_{rel} = 0$ or $k_{irr} = 0$, then the criteria are modified in the obvious ways: If $k_{rel} = 0$, then $E(\widehat{k}_{rel}) \to 0$ as $n \to \infty$, and if $k_{irr} = 0$, then $E(\widehat{k}_{irr}) \to 0$ as $n \to \infty$. The proportion of spuriously retained variables, $(\widehat{k}_{irr}/k_{irr})$ is also referred to as gauge in this literature, with distributional results on the gauge for a specific case (the variables being impulses as in IIS) provided in Johansen and Nielsen (2016).

GETS modelling combines well-known ingredients from the model-selection literature: Backwards elimination, tests on the β_j's (both single and multiple hypothesis tests), diagnostics tests and fit-measures (e.g. information criteria). Specifically, GETS modelling may be described as proceeding in three steps:

1. Formulate a General Unrestricted Model (GUM) that passes a set of chosen diagnostic tests.[2] Each non-significant regressor in the GUM constitutes the starting point of a backwards elimination path, and a regressor is non-significant if the p-value of a two-sided t-test is lower than the chosen significance level α.

2. Undertake backwards elimination along multiple paths by removing, one-by-one, non-significant regressors. Each removal is checked for validity against the chosen set of diagnostic tests, and for parsimonious encompassing (i.e. a multiple hypothesis test) against the GUM.

3. Select, among the terminal models, the specification with the best fit according to a fit-criterion, e.g. the Schwarz (1978) information criterion.

3. Setting time-series attributes

The **gets** package does not require that time-series characteristics are set beforehand. However, if they are not, and if the data are in fact time series, then graphs and other outputs (e.g. fitted values, residuals, etc.) are not optimal. The **gets** package is optimised to work with Zeileis's Ordered Observation (ZOO) package **zoo**, see Zeileis and Grothendieck (2005). In fact, the fitted values, residuals, recursive estimates and so on returned by **gets** functions, are all objects of class `zoo`. The **zoo** package provides a very general and versatile infrastructure for observations that are ordered according to an arbitrary index, e.g. time-series, and **zoo** is adapted to interact well with the less versatile time-series class of the **base** distribution, `ts`: To convert `ts` objects to `zoo` objects, simply use `as.zooreg` (preferred) or `as.zoo`.

3.1. Yearly, quarterly and monthly data

[2] Currently, the diagnostic tests available in **gets** are tests for serial correlation and ARCH in the standardised residuals, and a test for non-normality.

The most commonly used time-series indices are yearly, quarterly and monthly. In **zoo**, the recommended function to set these time-series attributes is `zooreg`, which is short for "regular" `zoo`-object. For example, the Hoover and Perez (1999) data, which is part of the **gets** package, can be loaded into the workspace by typing

```
data(hpdata)
```

To have a look at these data, type `head(hpdata)`, and for more details type `help(hpdata)`. The data are quarterly and run from 1959:1 to 1995:1. Using the `zooreg` function, we can set the quarterly time-series attributes as follows:

```
hpdata <- zooreg(hpdata[,-1], frequency=4, start=c(1959,1), end=c(1995,1))
```

Note that `[,-1]` removes the first column, i.e. the quarterly index, since it is not needed. Next, to plot all the series type `plot(hpdata)`. The variable in the dataset that has received the most attention is GCQ, which is US personal consumption. To have a look at this variable, type `hpdata[,"GCQ"]` or `plot(hpdata[,"GCQ"])`. The plot reveals that the variable is clearly non-stationary, since it is trending upwards over the whole sample. So it may be desirable to define a new and transformed variable, say, the log-difference in percent. This is achieved by

```
dlogcons <- diff(log(hpdata[,"GCQ"]))*100
```

To plot this series (in blue instead of the black default), we can use `plot(dlogcons, col="blue")`.

3.2. Weekly, daily and higher frequencies

To set daily and weekly time-series attributes, it is recommended to use the `zoo` function (instead of `zooreg`) in combination with the `Date` function (part of the **base** distribution). For example, to define the time-index for a daily Standard and Poor's 500 (SP500) series, then the following code can be used:[3]

```
sp500Data <- read.csv("http://www.sucarrat.net/R/gets/sp500.csv")
sp500Data <- zoo(sp500Data[,-1], order.by=as.Date(sp500Data[,"Date"]))
```

Again, `[,-1]` removes the first column (the dates). To view the first observations type `head(sp500Data)`, and to plot the data type `plot(sp500Data)`.

For intraday data, the `POSIXct` and `POSIXlt` functions are particularly useful in defining intraday indices, see the webpage of the **zoo** package for several short intros and vignettes: https://cran.r-project.org/package=zoo. Finally, a **zoo** variant that aims at facilitating the handling of intraday financial data, is the **xts** package, see Ryan and Ulrich (2014).

4. The AR-X model with log-ARCH-X errors

[3] The source of the data is Yahoo Finance: http://finance.yahoo.com/q/hp?s=^GSPC+Historical+Prices. Downloaded 9 March 2016.

The AR-X model with log-ARCH-X errors is made up of two equations, one for the mean and one for the log-variance:

$$y_t = \phi_0 + \sum_{r=1}^{R} \phi_r y_{t-r} + \sum_{s=1}^{S} \eta_s x_{s,t}^m + \epsilon_t, \qquad \epsilon_t = \sigma_t z_t, \quad z_t \sim iid(0,1), \tag{3}$$

$$\begin{aligned} \ln \sigma_t^2 = {} & \alpha_0 + \sum_{p=1}^{P} \alpha_p \ln \epsilon_{t-p}^2 + \sum_{q \in Q} \beta_q \ln EqWMA_{q,t-1} \\ & + \sum_{a=1}^{A} \lambda_a (\ln \epsilon_{t-a}^2) I_{\{\epsilon_{t-a}<0\}} + \sum_{d=1}^{D} \delta_d x_{d,t}^v. \end{aligned} \tag{4}$$

The conditional mean equation (3) is an Autoregressive (AR) specification of order R with S covariates $x_{1,t}^m, \ldots, x_{S,t}^m$ ("X"), AR-X for short. The error term ϵ_t is a product of the time-varying conditional standard deviation $\sigma_t > 0$ and the innovation $z_t \in \mathbb{R}$, where z_t is iid with zero mean and unit variance conditional on the past. The conditional log-variance equation (4) is given by a logarithmic Autoregressive Conditional Heteroscedasticity (log-ARCH) specification of order P with volatility proxies defined as $EqWMA_{q,t-1} = (\epsilon_{t-1}^2 + \cdots + \epsilon_{t-q}^2)/q$, A logarithmic asymmetry terms (*i.e.* "leverage") analogous to those of Glosten, Jagannathan, and Runkle (1993), and with D covariates $x_{1,t}^v, \ldots, x_{D,t}^v$, log-ARCH-X for short. The covariates in the mean need not be the same as those of the log-variance specification, hence the superscripts m and v, respectively. The log-proxies $\ln EqWMA_{q,t-1}$, where EqWMA is short for Equally Weighted Moving Average, are intended to proxy lagged log-GARCH terms, e.g. $\ln \sigma_{t-1}^2$. However, it should be noted that the log-proxies can also be given additional interpretation of interest. For example, if $y_t = \epsilon_t$ is a daily financial return, and if the returns are recorded over weekdays only, then $EqWMA_{5,t-1}$, $EqWMA_{20,t-1}$ and $EqWMA_{60,t-1}$ can be interpreted as the "weekly", "monthly" and "quarterly" volatilities, respectively. The log-proxies thus provide great flexibility in modelling the persistence of log-volatility. Also, note that $EqWMA_{q,t-1} = \ln \epsilon_{t-1}^2$, i.e. the ARCH(1) term, when $q = 1$. Of course, additional volatility proxies can be included via the covariates $x_{d,t}$.

The model (3)-(4) is estimated in two steps. First, the mean specification (3) is estimated by OLS. The default variance-covariance matrix is the ordinary one, but – optionally – this can be changed to either that of White (1980) or that of Newey and West (1987). Second, the non-linear AR-representation of (4) is estimated, also by OLS. The nonlinear AR-representation is given by

$$\ln \epsilon_t^2 = \alpha_0^* + \sum_{p=1}^{P} \alpha_p \ln \epsilon_{t-p}^2 + \sum_{q \in Q} \beta_q \ln EqWMA_{q,t-1} + \sum_{a=1}^{A} \lambda_a (\ln \epsilon_{t-a}^2) I_{\{\epsilon_{t-a}<0\}} + \sum_{d=1}^{D} \delta_d x_{d,t}^v + u_t,$$

where $\alpha_0^* = \alpha_0 + E(\ln z_t^2)$ and $u_t = \ln z_t^2 - E(\ln z_t^2)$ with $u_t \sim iid(0, \sigma_u^2)$. This provides consistent estimates of all the parameters in (4) except α_0, under appropriate assumptions. To identify α_0, an estimate of $E(\ln z_t^2)$ is needed, which depends on the density of z_t. Sucarrat, Grønneberg, and Escribano (2015) show that a simple formula made up of the residuals $\widehat{u}_t$ provides a consistent and asymptotically normal estimate under very general and non-restrictive assumptions. The estimator is essentially the negative of the natural log of the smearing estimate of Duan (1983): $\widehat{E(\ln z_t^2)} = -\ln\left[n^{-1}\sum_{t=1}^{n} \exp(\widehat{u}_t)\right]$. So the expression in square brackets is the smearing estimate. The log-variance intercept α_0 can thus be estimated

by $\widehat{\alpha}_0^* - \widehat{E(\ln z_t^2)}$. Finally, the ordinary variance-covariance matrix is used for inference in the log-variance specification, since the error term u_t of the nonlinear AR-representation is *iid*.

4.1. Simulation

Simulation from an AR(R) model can readily be done with the `arima.sim` function in the **stats** package (part of the base distribution of R). For example, the following code simulates 100 observations from the AR(1) model $y_t = \phi_0 + \phi_1 y_{t-1} + \epsilon_t$ with $\phi_0 = 0$ and $\phi_1 = 0.4$:

```
set.seed(123)
y <- arima.sim(list(ar = 0.4), 100)
```

To simulate from a model with log-ARCH errors, we first need to simulate the errors. This can be achieved with `lgarchSim` from the **lgarch** package (Sucarrat (2014)):

```
library(lgarch)
```

Next, the following code simulates an error-term ϵ_t that follows the log-ARCH(1) specification $\ln \sigma_t^2 = \alpha_0 + \alpha_1 \ln \epsilon_{t-1}^2$ with $\alpha_0 = 0$ and $\alpha_1 = 0.3$:

```
eps <- lgarchSim(100, arch=0.3, garch=0)
```

By default, the standardised error z_t is normal, but this can be changed via the `innovation` argument of the `lgarchSim` function. To combine the log-ARCH error with an AR(1) model with $\phi_0 = 0$ and $\phi_1 = 0.4$ the following code can be used:

```
yy <- arima.sim(list(ar=0.4), 100, innov=eps)
```

The command `plot(as.zoo(cbind(y,yy,eps)))` plots the three series.

4.2. `arx`: Estimation

The function `arx` estimates an AR-X model with log-ARCH-X errors. For example, the following code fits an AR(1) model to the mean of the series `y` generated above, and stores the results in an object called `mod01`:

```
mod01 <- arx(y, ar=1)
```

To print the estimation results, simply type `mod01`. This returns:

```
Date: Fri Feb 26 14:42:25 2016
Method: Ordinary Least Squares (OLS)
Variance-Covariance: Ordinary
No. of observations (mean eq.): 99
Sample: 2(1) to 100(1)

Mean equation:

         coef  std.error   t-stat     p-value
```

```
ar1 0.4001408 0.09450914 4.233884 5.18375e-05

Diagnostics:

                      Chi-sq  df     p-value
Ljung-Box AR(2)    0.2765043   2   0.8708791
Ljung-Box ARCH(1)  0.3169523   1   0.5734450
Jarque-Bera        0.1477797   2   0.9287740

R-squared          0.1522547
Log-lik.(n=99) -130.1268946
```

The three diagnostic tests are all of the standardised residuals $\widehat{z}_t$. The AR and ARCH tests are Ljung and Box (1979) tests for serial correlation in $\widehat{z}_t$ and $\widehat{z}_t^2$, respectively, and the number in parentheses indicates at which lag the test is conducted. The Jarque and Bera (1980) test is for non-normality. It should be noted though that normality of z_t is not required, neither for estimation nor for inference. `R-squared` is that of the mean specification, whereas the (Gaussian) log-likelihood is made up of the residuals $\widehat{\epsilon}_t$. If no log-variance specification is fitted, then the conditional variance in the log-likelihood is constant and equal to the sample variance of the residuals. By contrast, if a log-variance specification is fitted, then the conditional variance in the log-likelihood is equal to the fitted conditional variance, given by $\widehat{\sigma}_t^2 = \exp(\ln \widehat{\sigma}_t^2)$.

The main optional arguments of the `arx` function when estimating the mean are:

- `mc`: `TRUE` or `FALSE` (default). `mc` is short for "mean constant", so `mc = TRUE` includes an intercept, whereas `FALSE` does not.
- `ar`: integer vector that indicates the AR terms to include, say, `ar=1`, `ar=1:4` or `ar=c(2,4)`.
- `mxreg`: vector, matrix or `zoo` object that contains additional regressors to be included in the mean specification.
- `vcov.type`: the type of variance-covariance matrix used for inference in the mean specification. By default, the ordinary (`"ordinary"`) matrix is used. The other options available are `"white"`, i.e. the heteroscedasticity robust variance-covariance matrix of White (1980), and `"newey-west"`, i.e. the heteroscedasticity and autocorrelation robust variance-covariance matrix of Newey and West (1987).

To make full use of these arguments, let us first generate a set of 5 regressors:

```
mX <- matrix(rnorm(100*5), 100, 5)
```

Next, the following code estimates an AR-X model with an intercept, two AR-lags and five regressors, and stores the estimation results in an object called `mod02`:

```
mod02 <- arx(y, mc=TRUE, ar=1:2, mxreg=mX, vcov.type="white")
```

Estimation of the log-variance specification is also undertaken with the `arx` function. For example, the following code fits the log-ARCH(1) specification $\ln \sigma_t^2 = \alpha_0 + \alpha_1 \ln \epsilon_{t-1}^2$ to the variable `eps` generated above:

```
mod03 <- arx(eps, arch=1)
```

Typing `mod03` prints the estimation results. The main optional arguments when estimating the log-variance are:

- `arch`: integer vector that indicates the log-ARCH terms to include, say, `arch=1`, `arch=1:3` or `arch=c(3,5)`.
- `asym`: integer vector that indicates the logarithmic asymmetry terms (often referred to as "leverage") to include, say, `asym=1`, `asym=1:4`, or `asym=c(2,4)`.
- `vxreg`: vector, matrix or `zoo` object that contains additional regressors to be included in the log-volatility specification

The following code provides an example that makes use of all three arguments:

```
mod04 <- arx(eps, arch=1:3, asym=2, vxreg=log(mX^2))
```

Again, typing `mod04` prints the results. Finally we give an example where we jointly fit a mean and log-variance equation to the series `yy` generated above, using the variance-covariance matrix of White (1980) for the mean equation:

```
mod05 <- arx(yy, mc=TRUE, ar=1:2, mxreg=mX, arch=1:3, asym=2,
        vxreg=log(mX^2), vcov.type="white")
```

4.3. Extraction functions

Currently there are ten functions available for extracting information from `arx` objects. These functions (all S3 methods except `recursive`) are:

`coef`, `fitted`, `logLik`, `plot`, `predict`, `print`, `recursive`, `residuals`, `summary`, `vcov`

Six of these (`coef`, `fitted`, `predict`, `recursive`, `residuals` and `vcov`) have an optional argument that allows you to choose whether to extract information pertaining to the mean or log-variance specification. The `print` function prints the estimation result, `logLik` extracts the (Gaussian) log-likelihood associated with the joint model, `summary` lists the entries of the `arx` object (a `list`) and `plot` plots the fitted values and residuals of the model.

4.4. Example: A model of quarterly inflation with time-varying conditional variance

When Engle (1982) proposed the ARCH-class of models, his empirical application was the uncertainty of UK-inflation. However, the ARCH(4) specification he used to model the conditional variance was severely restricted in order to ensure the positivity of the variance estimates, see Engle (1982, p. 1002). Arguably, this is why (non-exponential) ARCH specifications never became popular in macroeconomics. The log-ARCH class of models, by contrast, does not suffer from the positivity problem, since the conditional variance is specified in logs. To illustrate we fit an AR(4)-X-log-ARCH(4)-X model to a quarterly inflation series, and

show that the conditional variance specification provides a substantial improvement in terms of fit and diagnostics.

The following code imports the data[4] and assigns it quarterly time-series attributes:

```
inflData <- read.csv("http://www.sucarrat.net/R/gets/inflation-quarterly.csv")
inflData <- zooreg(inflData[,-1], frequency=4, start=c(1989,1))
```

Note that `[,-1]` removes the first column, since it is not needed. The dataset thus contains four variables: *infl*, *q2dum*, *q3dum* and *q4dum*. The first variable is quarterly Norwegian inflation (year-on-year) in % from 1989(1) to 2015(4), whereas the latter three are seasonal dummies associated with the second, third and fourth quarter, respectively. Initially, to illustrate why a time-varying conditional variance is needed, we estimate only the mean specification:

$$infl_t = \phi_0 + \sum_{r=1}^{4} \phi_r infl_{t-r} + \eta_2 q2dum_t + \eta_3 q3dum_t + \eta_4 q4dum_t + \epsilon_t.$$

That is, an AR(4)-X, where the dummies constitute the X-part. The code

```
inflMod01 <- arx(inflData["infl"], mc=TRUE, ar=1:4, mxreg=inflData[,2:4],
        vcov.type="white")
```

estimates the model using heteroscedasticity-robust coefficient standard errors of the White (1980) type, and typing `inflMod01` prints the estimation results:

```
Date: Fri Feb 26 15:02:57 2016
Method: Ordinary Least Squares (OLS)
Variance-Covariance: White (1980)
No. of observations (mean eq.): 104
Sample: 1990(1) to 2015(4)

Mean equation:

               coef std.error      t-stat      p-value
mconst  0.838631074 0.2961338  2.83193261 5.637500e-03
ar1     0.725755002 0.1300407  5.58098556 2.211243e-07
ar2     0.019591100 0.1171347  0.16725278 8.675230e-01
ar3     0.035009234 0.1385735  0.25264010 8.010865e-01
ar4    -0.167675074 0.1336972 -1.25414030 2.128362e-01
q2dum  -0.014889213 0.2333917 -0.06379496 9.492661e-01
q3dum  -0.007297164 0.2262704 -0.03224975 9.743398e-01
q4dum   0.010399039 0.2226772  0.04670006 9.628493e-01

Diagnostics:
```

[4] The source of the data is Statistics Norway. The original untransformed data, a monthly Consumer Price Index (CPI), was retrieved 14 February 2016 from http://www.ssb.no/tabell/08183/.

```
                     Chi-sq df        p-value
Ljung-Box AR(5)   16.320533  5   0.0059860979
Ljung-Box ARCH(1)  5.966493  1   0.0145802489
Jarque-Bera       14.350354  2   0.0007653503

R-squared          0.5316566
Log-lik.(n=104) -110.4145511
```

The diagnostics suggest the standardised residuals are autocorrelated and heteroscedastic, since the tests for autocorrelation and heteroscedasticity yield p-values of 0.6% and 1.5%, respectively. Next, we specify the conditional variance as a log-ARCH(4)-X, where the X-part is made up of the seasonal dummies:

$$\ln \sigma_t^2 = \alpha_0 + \sum_{p=1}^{4} \alpha_p \ln \epsilon_{t-p}^2 + \delta_2 q2dum_t + \delta_3 q3dum_t + \delta_4 q4dum_t.$$

The code

```
inflMod02 <- arx(inflData[,"infl"], mc=TRUE, ar=1:4, mxreg=inflData[,2:4],
        arch=1:4, vxreg=inflData[,2:4], vcov.type="white")
```

estimates the full model with White (1980) standard errors in the mean and ordinary standard errors in the log-variance. Typing `inflMod02` returns

```
Date: Fri Feb 26 15:03:57 2016
Method: Ordinary Least Squares (OLS)
Variance-Covariance: White (1980)
No. of observations (mean eq.): 104
No. of observations (variance eq.): 100
Sample: 1990(1) to 2015(4)

Mean equation:

               coef std.error      t-stat      p-value
mconst  0.838631074 0.2961338  2.83193261 5.637500e-03
ar1     0.725755002 0.1300407  5.58098556 2.211243e-07
ar2     0.019591100 0.1171347  0.16725278 8.675230e-01
ar3     0.035009234 0.1385735  0.25264010 8.010865e-01
ar4    -0.167675074 0.1336972 -1.25414030 2.128362e-01
q2dum  -0.014889213 0.2333917 -0.06379496 9.492661e-01
q3dum  -0.007297164 0.2262704 -0.03224975 9.743398e-01
q4dum   0.010399039 0.2226772  0.04670006 9.628493e-01

Log-variance equation:

              coef std.error     t-stat     p-value
vconst  0.95935218 0.5346362  3.2198780 0.072749054
```

```
arch1   0.16697428 0.1035178  1.6130011 0.110169438
arch2   0.12026920 0.1033531  1.1636724 0.247565919
arch3   0.14739794 0.1033161  1.4266691 0.157060220
arch4   0.05982037 0.1051547  0.5688797 0.570823925
q2dum  -1.32859656 0.6186157 -2.1476929 0.034366312
q3dum  -0.92706844 0.5840008 -1.5874438 0.115843277
q4dum  -1.82735554 0.6201431 -2.9466673 0.004069453

Diagnostics:

                     Chi-sq df     p-value
Ljung-Box AR(5)   9.1775879  5   0.1021870
Ljung-Box ARCH(5) 1.7613164  5   0.8810862
Jarque-Bera       0.1283933  2   0.9378206

R-squared         0.5316566
Log-lik.(n=100) -82.3289236
```

The first noticeable difference between `inflMod01` and `inflMod02` is that the diagnostics improve substantially. In `inflMod02`, the AR and ARCH tests of the standardised residuals suggest the standardised error z_t is uncorrelated and homoscedastic at the usual significance levels (1%, 5% and 10%), and the Jarque and Bera (1980) test suggests z_t is normal. The second noticeable improvement is in terms of fit, as measured by the average (Gaussian) log-likelihood. In `inflMod01` the average log-likelihood is $-110.4146/104 = -1.06$, whereas in `inflMod02` the average log-likelihood is $-82.3289/100 = -0.82$. This is a substantial increase. In terms of the Schwarz (1978) information criterion (SC), for example, which favours parsimony, the value falls from 2.53 in `inflMod01`, computed as `info.criterion(as.numeric(logLik(inflMod01)), n=104, k=8+1)`, to 2.38 in `inflMod02`, computed as `info.criterion(as.numeric(logLik(inflMod02)), n=100, k=8+8)`. Together, the enhanced fit and diagnostics means the log-variance specification provides a notable improvement. Later, in Section 5.4, we will undertake GETS modelling of the mean and variance specifications of `inflMod02`.

4.5. Example: A rich model of daily SP500 volatility

The most common volatility specification in finance are first order GARCH-like specifications. In the log-GARCH class of models, this corresponds to a log-GARCH(1,1): $\ln \sigma_t^2 = \alpha_0 + \alpha_1 \ln \epsilon_{t-1}^2 + \beta_1 \ln \sigma_{t-1}^2$. Here, we show that a log-ARCH-X model that makes use of commonly available information provides a better fit.

In Section 3.2 we loaded a dataset of the Standard and Poor's 500 (SP500) index that was named `sp500Data`. The dataset contains the daily value of the SP500 index, its highs and lows, and daily volume. We will make use of this information together with day-of-the-week dummies to construct a rich model of SP500 return volatility. But first we shorten the sample, since not all variables are available from the start:

```
sp500Data <- window(sp500Data, start=as.Date("1983-07-01"))
```

The resulting sample thus goes from 1 July 1983 to 8 March 2016, a total of 8241 observations

before differencing and lagging. Next, the following lines of code makes the log-return in percent, a lagged range-based volatility proxy, and the lagged log-difference of volume:

```
##make log-returns in %:
sp500Ret <- diff(log(sp500Data[,"Adj.Close"]))*100

##make lagged volatility proxy (range-based):
relrange <- (log(sp500Data[,"High"]) - log(sp500Data[,"Low"]) )*100
volproxy <- log(relrange^2)
volproxylag <- lag(volproxy, k=-1)

##make volume variable:
volume <- log(sp500Data[,"Volume"])
volumediff <- diff(volume)*100
volumedifflag <- lag(volumediff, k=-1)
```

Finally, we make the day-of-the-week dummies and estimate the full model, a log-ARCH(5)-X specification:

```
##make day-of-the-week dummies:
sp500Index <- index(sp500Ret)
days <- weekdays(sp500Index)
days <- union(days,days)
dTue <- zoo(as.numeric(weekdays(sp500Index)==days[1]),
        order.by=sp500Index)
dWed <- zoo(as.numeric(weekdays(sp500Index)==days[2]),
        order.by=sp500Index)
dThu <- zoo(as.numeric(weekdays(sp500Index)==days[3]),
        order.by=sp500Index)
dFri <- zoo(as.numeric(weekdays(sp500Index)==days[4]),
  order.by=sp500Index)

##estimate log-arch(5)-x:
sp500Mod01 <- arx(sp500Ret, arch=1:5, log.ewma=c(5,20,60,120),
        asym=1, vxreg=cbind(volproxylag,volumedifflag,dTue,dWed,dThu,dFri))
```

Typing `sp500Mod01` returns the following print:

```
Date: Fri Mar 11 20:40:54 2016
Method: Ordinary Least Squares (OLS)
No. of observations (variance eq.): 8235
Sample: 1983-07-05 to 2016-03-08

Log-variance equation:

                     coef   std.error        t-stat      p-value
vconst        -0.001139917 0.077897180  0.0002141426 9.883245e-01
arch1         -0.046889672 0.016096095 -2.9131085694 3.588142e-03
```

```
arch2          0.003586491 0.012149216  0.2952035148 7.678459e-01
arch3          0.024878438 0.012174758  2.0434440879 4.104032e-02
arch4          0.013292127 0.012134133  1.0954328047 2.733592e-01
arch5          0.036940556 0.012202167  3.0273766359 2.474511e-03
asym1         -0.032861516 0.017387150 -1.8899886094 5.879464e-02
logEqWMA(5)    0.027100861 0.051681990  0.5243772643 6.000303e-01
logEqWMA(20)   0.286593626 0.071160714  4.0274135859 5.690127e-05
logEqWMA(60)   0.203310293 0.105046413  1.9354329896 5.297145e-02
logEqWMA(120)  0.192228937 0.086451191  2.2235545168 2.620551e-02
volproxylag    0.199830207 0.039783155  5.0229853771 5.194553e-07
volumedifflag -0.003117807 0.001413439 -2.2058307616 2.742343e-02
dTue           0.106419362 0.082816983  1.2849944284 1.988304e-01
dWed          -0.059534064 0.084723469 -0.7026868039 4.822709e-01
dThu           0.087642610 0.083756975  1.0463917791 2.954110e-01
dFri           0.082379884 0.083380874  0.9879949639 3.231842e-01

Diagnostics:

                        Chi-sq df      p-value
Ljung-Box AR(1)   7.142086e-01  1 3.980502e-01
Ljung-Box ARCH(6) 3.063395e+01  6 2.977139e-05
Jarque-Bera       1.876561e+04  2 0.000000e+00

R-squared             0.00
Log-lik.(n=8235) -11128.55
```

Later, in Section 5.5, we will simplify this model with the `getsv` function. For now, we provide a comparison with a log-GARCH(1,1) using the R package **lgarch**, see Sucarrat (2014). The following code loads the package and estimates the model:

```
library(lgarch)
sp500Mod02 <- lgarch(sp500Ret)
```

Extracting the log-likelihood by `logLik(sp500Mod02)` reveals that it is substantially lower, namely −11396.11. In terms of the Schwarz (1978) information criterion, the value increases from 2.72 in `sp500Mod01`, computed as `info.criterion(as.numeric(logLik(sp500Mod01)), n=8235, k=17)`, to 2.77 in `sp500Mod02`, computed as `info.criterion(as.numeric(logLik(sp500Mod02)), n=8240, k=3)`.

5. GETS modelling

5.1. getsm: Modelling the mean

GETS modelling of the mean specification is undertaken by applying the `getsm` function on an `arx` object. For example, the following code performs GETS model selection of the mean specification of `mod05` with default values on all the optional arguments:

```
getsm05 <- getsm(mod05)
```

The results are stored in an object named `getsm05`, and typing `getsm05` gives:

```
Date: Fri Feb 26 15:28:01 2016
Method: Ordinary Least Squares (OLS)
Variance-Covariance: White (1980)
No. of observations (mean eq.): 98
Sample (mean eq.): 3 to 100

GUM mean equation:

       reg.no keep         coef  std.error      t-stat    p-value
mconst      1    0 -0.059689419 0.07822847 -0.76301399 0.44745043
ar1         2    0  0.193815687 0.12354564  1.56877801 0.12020902
ar2         3    0  0.034380293 0.11415593  0.30116958 0.76397990
mxreg1      4    0  0.117104503 0.08058382  1.45320118 0.14964589
mxreg2      5    0  0.011612423 0.08659254  0.13410420 0.89361961
mxreg3      6    0 -0.108716187 0.08159458 -1.33239477 0.18609432
mxreg4      7    0 -0.222672205 0.10198196 -2.18344698 0.03160427
mxreg5      8    0  0.001249764 0.06940237  0.01800751 0.98567273

GUM log-variance equation:

              coef  std.error     t-stat    p-value
vconst  0.35187159 0.43868736  0.6433661 0.42249451
arch1   0.26897541 0.10747036  2.5027870 0.01423593
arch2   0.08853989 0.15913476  0.5563831 0.57941084
arch3   0.02293221 0.11586060  0.1979293 0.84357298
asym2  -0.11294120 0.17176712 -0.6575251 0.51261998
vxreg1  0.10218147 0.11037371  0.9257772 0.35718279
vxreg2 -0.06887269 0.09376217 -0.7345466 0.46463728
vxreg3 -0.03200568 0.10259747 -0.3119539 0.75583946
vxreg4  0.02942943 0.10686462  0.2753899 0.78368507
vxreg5  0.18717555 0.12025907  1.5564361 0.12332011

Diagnostics:

                     Chi-sq df    p-value
Ljung-Box AR(3)   0.1867169  3 0.97970462
Ljung-Box ARCH(4) 0.4398348  4 0.97908752
Jarque-Bera       7.3949185  2 0.02478642

Paths searched:

path 1 : 1 8 5 3 4 6 2
path 2 : 2 8 5 3 1 4 6
```

```
path 3 : 3 8 5 1 4 6 2
path 4 : 4 3 5 8 1 6 2
path 5 : 5 8 3 1 4 6 2
path 6 : 6 8 5 3 1 4 2
path 7 : 8 5 3 1 4 6 2

Terminal models:

spec 1 : 1 2 3 4 5 6 7 8
spec 2 : 7

              info(sc)      logl  n k
spec 1 (gum): 2.757982 -112.7887 95 8
spec 2:       2.357648 -109.7113 95 1

SPECIFIC mean equation:

             coef std.error    t-stat    p-value
mxreg4 -0.2545617 0.0992512 -2.564823 0.01185581

SPECIFIC log-variance equation:

              coef  std.error     t-stat     p-value
vconst  0.39548049 0.37969332  1.0848862 0.297607035
arch1   0.32608814 0.10674775  3.0547543 0.003008216
arch2   0.09771601 0.14937485  0.6541664 0.514770643
arch3   0.07132929 0.10693034  0.6670632 0.506538601
asym2  -0.05431815 0.15673108 -0.3465691 0.729771538
vxreg1  0.19150223 0.08728681  2.1939425 0.030968647
vxreg2  0.04951203 0.07578560  0.6533171 0.515315217
vxreg3 -0.07403269 0.08369566 -0.8845463 0.378896643
vxreg4 -0.03463747 0.08407657 -0.4119752 0.681394662
vxreg5  0.01700620 0.09657184  0.1760990 0.860635062

Diagnostics:

                     Chi-sq df   p-value
Ljung-Box AR(3)   2.0304870  3 0.5661032
Ljung-Box ARCH(4) 6.9536895  4 0.1383559
Jarque-Bera       0.1002355  2 0.9511174
```

The first part of the printed results pertains to the GUM. Note in particular that regressors are numbered (the *reg.no* column) in the *GUM mean equation*. This is useful when interpreting *Paths searched*, which indicates in which order the regressors are deleted in each path. Next, the *Terminal models* part lists the distinct terminal specifications. Note that the GUM is always included in this list to ensure a non-empty list. By default, the Schwarz (1978) information criterion (sc) is used to choose among the terminals, but this can be changed (see

below). The last part contains the estimation results of the final, simplified model.

The main optional arguments of the `getsm` function are (type `args(getsm)` or `?getsm` for all the arguments):

- `t.pval`: numeric value between 0 and 1 (The default is 0.05). The significance level used for the two-sided t-tests of the regressors.
- `wald.pval`: numeric value between 0 and 1 (the default is `t.pval`). The significance level used for the Parsimonious Encompassing Test (PET) against the General Unrestricted Model (GUM) at each regressor deletion.
- `do.pet`: logical, `TRUE` (the default) or `FALSE`. If `TRUE`, then a PET against the GUM is undertaken at each regressor removal.
- `ar.LjungB`: a list with two elements named `lag` and `pval`, respectively, or `NULL`. If the list is not `NULL`, then a Ljung and Box (1979) test for serial correlation in the standardised residuals is undertaken at each attempt to remove a regressor. The default, `list(lag=NULL, pval=0.025)`, means the lag is chosed automatically (as `max(ar)+1`), and that a p-value of `pval=0.025` is used. If the list is `NULL`, then the standardised residuals $\widehat{z}_t$ are not checked for serial correlation after each removal.
- `arch.LjungB`: a list with two elements named `lag` and `pval`, respectively, or `NULL`. If the list is not `NULL`, then a Ljung and Box (1979) test for serial correlation in the *squared* standardised residuals is undertaken at each attempt to remove a regressor. The default, `list(lag=NULL, pval=0.025)`, means the lag is chosed automatically (as `max(arch)+1`) and that a p-value of `pval=0.025` is used. If the list is `NULL`, then the *squared* standardised residuals $\widehat{z}_t^2$ are not checked for serial correlation after each removal.
- `vcov.type`: `NULL`, `"ordinary"`, `"white"` or `"newey-west"`. If `NULL` (default), then the type of variance-covariance matrix is automatically determined (the option from the `arx` object is used). If `"ordinary"`, then the ordinary variance-covariance matrix is used. If `"white"`, then the variance-covariance matrix of White (1980) is used. If `"newey-west"`, then the variance-covariance matrix of Newey and West (1987) is used.
- `keep`: either `NULL` or an integer vector. If `NULL` (default), then no regressors are excluded from removal. Otherwise, the regressors associated with the numbers in `keep` are excluded from the removal space. For example, `keep=c(1)` excludes the intercept from removal. Retaining variables using the `keep` argument implements the "theory-embedding" approach outlined in Hendry and Johansen (2015) by "forcing" theory variables to be retained while conducting model discovery beyond the set of forced variables.
- `info.method`: `"sc"`, `"aic"` or `"hq"`. If `"sc"` (default), then the information criterion of Schwarz (1978) is used as tiebreaker between the terminals. If `"aic"`, then the information criterion of Akaike (1974) is used, and if `"hq"`, then the information criterion of Hannan and Quinn (1979) is used

As an example, the following code uses a lower significance level for the regressor significance tests and the PETs, and turns of diagnostic testing for ARCH in the standardised residuals:

```
getsm05a <- getsm(mod05, t.pval=0.01, arch.LjungB=NULL)
```

Similarly, the following code restricts the mean intercept from being deleted, even though it is not significant:

```
getsm05b <- getsm(mod05, keep=c(1))
```

5.2. getsv: Modelling the log-variance

GETS modelling of the log-variance specification is undertaken by applying the `getsv` function to an `arx` object. For example, the following code performs GETS model selection of the log-variance specification of `mod05` with default values on all the optional arguments:

```
getsv05 <- getsv(mod05)
```

Alternatively, the following code undertakes GETS model selection on the log-variance specification of the simplified model `getsm05`:

```
mod06 <- arx(residuals(getsm05), arch=1:3, asym=2, vxreg=log(mX^2))
getsv06 <- getsv(mod06)
```

Typing `getsv06` prints the results, whose structure is organised in a similar way to those returned by `getsm` (see above). Note, though, that *vconst*, the log-variance intercept, is forced to enter the `keep` set when `getsv` is used. That is, α_0 is rectricted from removal even if it is not significant. This is due to the estimation procedure. Finally, the main optional arguments of `getsv` are almost the same as those of `getsm` (see above). The main difference is that the only variance-covariance matrix available is the ordinary one, since the error-term of the AR-specification is *iid*. As an example of how to set some of the options to non-default values, the following code restricts the three log-ARCH terms (in addition to the log-variance intercept) from removal, and turns off diagnostic testing for serial correlation in the standardised residuals:

```
getsv06b <- getsv(mod06, keep=1:4, ar.LjungB=NULL)
```

5.3. Extraction functions

There are ten extraction functions available for `gets` objects, i.e. objects produced by either `getsm` or `getsv`. These functions (seven of them S3 methods) are:

`coef`, `fitted`, `paths`, `plot`, `print`, `recursive`, `residuals`, `summary`, `terminals`, `vcov`

All, apart from `paths` and `terminals`, behave in a similar way to the corresponding extraction functions for `arx` objects. In particular, `coef`, `fitted`, `print` and `residuals` automatically detect whether `getsm` or `getsv` has been used, and behaves accordingly. The `paths` function extracts the paths searched, and `terminals` the terminal models.

5.4. Example: A parsimonious model of quarterly inflation

In Section 4.4, we showed that a log-ARCH(4)-X specification of the log-variance improved the fit and diagnostics of an AR(4)-X model of quarterly inflation. Here, we will obtain a simplified version by using the `getsm` and `getsv` functions.

The estimation results of the AR(4)-X-log-ARCH(4)-X specification that we fitted was stored as an `arx` object named `inflMod02`. The following code undertakes GETS modelling of the mean, and stores the results in an object named `inflMod03`:

```
inflMod03 <- getsm(inflMod02)
```

Next, typing `inflMod03` prints the results (for brevity, only selected parts are reproduced):

```
GUM mean equation:

       reg.no keep         coef std.error      t-stat      p-value
mconst      1    0  0.838631074 0.2961338  2.83193261 5.637500e-03
ar1         2    0  0.725755002 0.1300407  5.58098556 2.211243e-07
ar2         3    0  0.019591100 0.1171347  0.16725278 8.675230e-01
ar3         4    0  0.035009234 0.1385735  0.25264010 8.010865e-01
ar4         5    0 -0.167675074 0.1336972 -1.25414030 2.128362e-01
q2dum       6    0 -0.014889213 0.2333917 -0.06379496 9.492661e-01
q3dum       7    0 -0.007297164 0.2262704 -0.03224975 9.743398e-01
q4dum       8    0  0.010399039 0.2226772  0.04670006 9.628493e-01

Paths searched:

path 1 : 3 7 6 8 4 5 -5
path 2 : 4 7 6 8 3 5 -5
path 3 : 5 7 6 3 8 -8 4 -4
path 4 : 6 7 8 3 4 5 -5
path 5 : 7 6 8 3 4 5 -5
path 6 : 8 7 6 3 4 5 -5

Terminal models:

spec 1 : 1 2 3 4 5 6 7 8
spec 2 : 1 2 5
spec 3 : 1 2 4 8

              info(sc)      logl   n k
spec 1 (gum): 2.014992 -82.32892 100 8
spec 2:       1.790069 -82.59571 100 3
spec 3:       1.845766 -83.07798 100 4

SPECIFIC mean equation:

             coef  std.error    t-stat      p-value
mconst  0.8415598 0.20103040  4.186232 6.068307e-05
```

```
ar1     0.7490429 0.10190207  7.350615 5.272873e-11
ar4    -0.1390594 0.09899779 -1.404672 1.631869e-01
```

The final model contains the AR(1) and AR(4) terms, but no quarterly dummies. So the level of quarterly year-on-year inflation does not seem to depend on quarter. Note that, in *Paths searched*, regressor no. 5 (i.e. the AR(4) term) has a minus sign in front of it in all but one of the searched paths. This means the term has been re-introduced after deletion, since its deletion leads to a violation of one or several of the diagnostics tests. This is the reason the AR(4) term is retained even though it is not significant in the final model. Next, we use the residuals of the simplified model to develop a parsimonious model of the log-variance, storing the results in `inflMod05`:

```
inflMod04 <- arx(residuals(inflMod03), arch=1:4, vxreg=inflData[,2:4])
inflMod05 <- getsv(inflMod04, ar.LjungB=list(lag=5,pval=0.025))
```

Note that, to ensure that the diagnostic test for autocorrelation in the standardised residuals is undertaken at the same lag as earlier, the `ar.LjungB` argument is modified. Next, typing `inflMod05` prints the results, and again we only reproduce selected parts in the interest of brevity:

```
SPECIFIC log-variance equation:

             coef std.error    t-stat     p-value
vconst  0.7131117 0.5396473  1.746204 0.186354623
arch1   0.1743751 0.1005689  1.733886 0.086216864
arch2   0.1682185 0.1003427  1.676440 0.096974964
q2dum  -1.4383418 0.6299229 -2.283362 0.024661705
q3dum  -1.0918880 0.6003535 -1.818742 0.072135106
q4dum  -1.8283612 0.6035097 -3.029547 0.003162664
```

The results suggest a high impact of the ARCH(1) and ARCH(2) terms – much higher than for financial returns,[5] and that the conditional variance is dependent on quarter. To get an idea of the economic importance of our results, we re-estimate the full, simplified model, and generate out-of-sample forecasts of the conditional standard deviation up to four quarters ahead. The full, simplified model re-estimated:

```
inflMod06 <- arx(inflData[,"infl"], ar=c(1,4), arch=1:2, vxreg=inflData[,2:4],
        vcov.type="white")
```

In order to generate out-of-sample forecasts, we first need to generate the out-of-sample values of the retained quarterly dummies:

```
newvxreg=matrix(0,4,3)
colnames(newvxreg) <- c("q2dum","q3dum","q4dum")
newvxreg[2,"q2dum"] <- 1
newvxreg[3,"q3dum"] <- 1
newvxreg[4,"q4dum"] <- 1
```

[5] In finance, if ϵ_t is a mean-corrected financial return, then the ARCH(1) term is usually about 0.05, and almost never higher than 0.1.

We can now generate the out-of-sample forecasts of the conditional standard deviations:

```
set.seed(123) #for reproducability
predict(inflMod06, n.ahead=4, spec="variance", newvxreg=newvxreg)
```

The first command, `set.seed(123)`, is for reproducability purposes, since a bootstrap procedure is used to generating variance forecasts two or more steps ahead (the number of draws can be changed via the `n.sim` argument). The forecasts for 2016(1) to 2016(4) are:

```
        1         2         3         4
1.0448239 0.3453098 0.4712113 0.2101471
```

In other words, the conditional variance is forecasted to be four times higher in 2016(1) than in 2016(4). This has notable economic consequences. For example, if the forecasted inflation in 2016(1) is 2%, then an approximate 95% prediction interval computed as $2 \pm 2 \times \widehat{\sigma}_{n+1}$ is given by the range 0% to 4%, which is large. By contrast, an approximate 95% prediction interval for 2016(4) computed as $2 \pm 2 \times \widehat{\sigma}_{n+4}$ is given by the range 1.1% to 2.9%, which is much tighter.

5.5. Example: A parsimonious model of daily SP500 volatility

In Section 4.5 we estimated a rich model of daily SP500 return volatility named `sp500Mod01`. Simplification of this model is staightforward with the `getsv` function. Nevertheless, since the model does not fully get rid of the ARCH in the standardised residuals, we will turn off the ARCH diagnostics. Also, for parsimony we will choose a small regressor significance level equal to 0.1%:

```
sp500Mod03 <- getsv(sp500Mod01, t.pval=0.001, arch.LjungB=NULL)
```

Typing `sp500Mod03` returns, towards the end, the following:

```
SPECIFIC log-variance equation:

                     coef   std.error    t-stat      p-value
vconst        -0.059212734 0.041007082  2.085031 1.487492e-01
arch1         -0.068339586 0.013729996 -4.977393 6.575438e-07
logEqWMA(5)    0.104930977 0.040915194  2.564597 1.034716e-02
logEqWMA(20)   0.363111087 0.060515809  6.000268 2.053243e-09
logEqWMA(120)  0.332407042 0.052687148  6.309073 2.952150e-10
volproxylag    0.194830384 0.039526750  4.929077 8.423348e-07
volumedifflag -0.003820265 0.001348062 -2.833894 4.609648e-03

Diagnostics:

                        Chi-sq df      p-value
Ljung-Box AR(1)       1.196839  1 0.2739544181
Ljung-Box ARCH(6)    24.854798  6 0.0003632495
Jarque-Bera       17160.911768  2 0.0000000000
```

In other words, no day-of-the-week dummies are retained and only the first ARCH-term is retained. However, three of the log-proxies are retained, i.e. the weekly, the monthly and the half-yearly, and both the lagged range-based volatility proxy and the lagged log-volume difference are retained. The log-likelihood is now −11131.4, which yields a Schwarz (1978) information criterion value equal to 2.71 (computed as `info.criterion(as.numeric(logLik(sp500Mod03)), n=8235, k=7)`).

6. Indicator saturation

Indicator saturation has been a crucial development in GETS modelling to address the distorting influence of outliers and structural breaks (changes in parameters) in econometric models. Such parameter changes are generally of unknown magnitudes and may occur at unknown times. Indicator saturation tackles this challenge by starting from a general model allowing for an outlier or shift at every point and removing all but significant ones using general-to-specific selection. This serves both as a method to detect outliers and breaks, as well as a generalised approach to model mis-specification testing – if the model is well-specified, then no outliers/shifts will be detected. The function `isat` conducts multi-path indicator saturation to detect outliers and location-shifts in regression models using impulse indicator saturation (IIS - see Hendry *et al.* 2007, and Johansen and Nielsen 2016 for a comprehensive asymptotic analysis), step-indicator saturation (SIS - see Castle *et al.* 2015), trend-indicator saturation (TIS - as applied in Pretis, Mann, and Kaufmann 2015), and user-designed indicator saturation (UIS, or designed break functions in Pretis, Schneider, Smerdon, and Hendry 2016). Formulating the detection of structural breaks as a problem of model selection, a regression model is saturated with a full set of indicators which are then selected over using the general-to-specific `getsm` algorithm at a chosen level of significance `t.pval`. This approach to break detection imposes no minimum break length, and outliers can be identified jointly with structural breaks. The respective GUMs for a simple model of the mean of y_t using impulse and step-indicator saturation[6] are given by equations (6) and (5):

$$\text{SIS GUM: } y_t = \mu + \sum_{j=2}^{T} \delta_j 1_{\{t \geq j\}} \tag{5}$$

$$\text{IIS GUM: } y_t = \mu + \sum_{j=1}^{T} \delta_j 1_{\{t = j\}} \tag{6}$$

where T denotes the total number of observations in the sample. Indicators are partitioned into blocks based on the `ratio.threshold` and `max.block.size`, where the block size used is the maximum of given by either criterion. Indicators retained in each block are re-combined and selected over to yield terminal models. Additional regressors that are not selected over can be included either through `mxreg` or `ar` specification. The different regimes made up of indicators (e.g. retained step-functions or impulses) weighted by their estimated coefficients describe shifts in the intercept over time – the coefficient path of the intercept.

[6] Note that specifications of step-functions are possible in SIS. Here we specify the steps as in equation (5), and thus for interpretation every additional step is added to the previous ones. In contrast, the paper introducing SIS (Castle *et al.* 2015) works with step-indicators of the form $\sum_{j=2}^{T} \delta_j 1_{\{t \leq j\}}$, in which case the steps have to be subtracted from the previous sum of steps to interpret the coefficients.

The primary arguments for selection of indicators in `isat` carry over from the `getsm` function. The main additional arguments are:

- `t.pval`: numeric value between 0 and 1. The significance level α used for the two-sided t-tests of the indicators in selection. The default is lower than in regular `getsm` model selection and set to 0.005 to control the number of false positives. Under the null of no outliers (or structural breaks), the irrelevance proportion or gauge (or proportion of spuriously retained indicators) is equal to αK where K is the number of indicators selected over. Thus setting $\alpha \approx 1/K$ yields one spuriously retained indicator on average under the null.
- `iis`: logical, `TRUE` or `FALSE`. If `TRUE`, then a full set of impulse indicators is added and selected over.
- `sis`: logical, `TRUE` or `FALSE`. If `TRUE`, then a full set of step indicators is added and selected over.
- `tis`: logical, `TRUE` or `FALSE`. If `TRUE`, then a full set of trend indicators is added and selected over.
- `uis`: matrix object that contains designed break functions to be selected over.
- `ratio.threshold`: numeric, between 0 and 1. Minimum ratio of variables in each block to total observations to determine the block size, default=0.8. Block size used is the maximum of given by either the ratio.threshold and max.block.size.
- `max.block.size`: an integer of at least 2. Maximum size of block of variables to be selected over, default=30. Block size used is the maximum of given by either the ratio.threshold and max.block.size.

6.1. Example: structural breaks in the growth rate of UK SO_2 emissions

Annual emissions of the pollutant sulphur dioxide (SO_2) in the UK have declined in the latter half of the 20th century due to policy interventions and changes in energy production. Here we assess whether there have been significant shifts in the growth rate ($\Delta log(SO_2)_t$) of sulphur dioxide emissions between 1946 and 2005, using the emission time series compiled by Smith, Aardenne, Klimont, Andres, Volke, and Delgado Arias (2011). Setting `t.pval` to 0.01 yields an approximate gauge of $0.01K$ under the null hypothesis of no shifts for K spuriously included variables. For IIS and SIS under the null, inclusion of a full set of indicators implies that $K = T$ for IIS, and $K = T - 1$ for SIS, and thus $0.01T = 0.01 \times 60$. This suggests less than one indicator being retained spuriously on average under the null of no shifts or outliers. Estimating an `isat` model using SIS:

```
so2 <- read.csv("http://www.sucarrat.net/R/gets/so2.csv")
yso2 <- zoo(so2[,"DLuk_tot_so2"], so2[,"year"])
sis <- isat(yso2, t.pval=0.01)

...
```

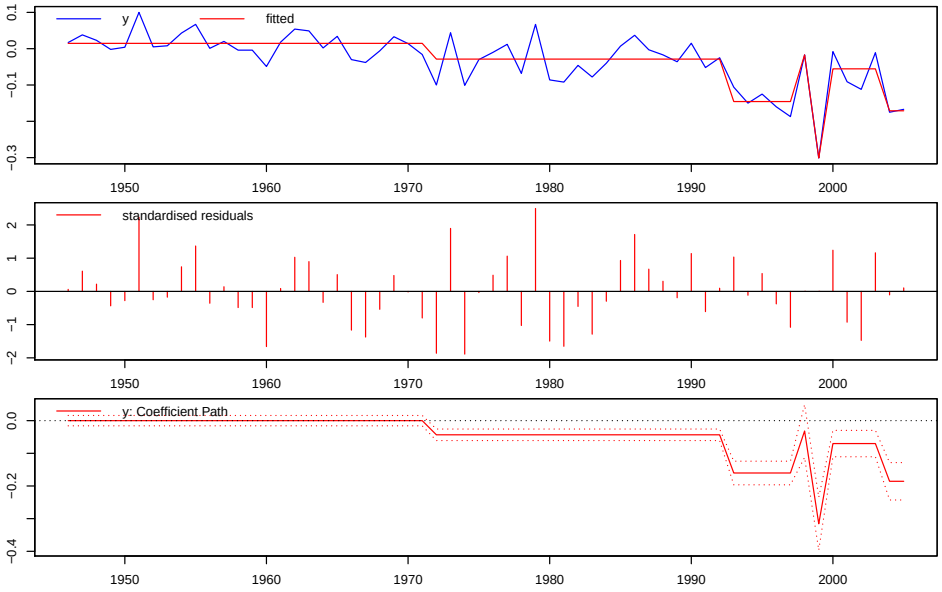
```
Searching path no. 25 out of 26
Searching path no. 26 out of 26

GETS of union of retained SIS indicators...

Searching path no. 1 out of 2
Searching path no. 2 out of 2

GETS of union of ALL retained indicators...

SPECIFIC mean equation:

               coef   std.error    t-stat      p-value
mconst   0.01465385 0.007931984  1.847438 7.026836e-02
sis1972 -0.04332051 0.011866458 -3.650669 5.990412e-04
sis1993 -0.11693333 0.020126141 -5.810023 3.625832e-07
sis1998  0.12860000 0.044305650  2.902564 5.382516e-03
sis1999 -0.28400000 0.057198348 -4.965178 7.505854e-06
sis2000  0.24550000 0.045219264  5.429102 1.441154e-06
sis2004 -0.11550000 0.035026692 -3.297485 1.746083e-03

Diagnostics:

                     Chi-sq df   p-value
Ljung-Box AR(1)   0.6155255  1 0.4327149
Ljung-Box ARCH(1) 1.4415297  1 0.2298920
Jarque-Bera       0.5730184  2 0.7508802
```

The above output shows multiple detected step-shifts (labelled `sis1972` – `sis2004`) in the time series. By default (`plot=TRUE`), `isat` also displays the output as in Figure 1 plotting the observed and fitted values, together with the coefficient path (the time-varying intercept through the regimes detected using SIS) as well as the standardised residuals. There is a downward step-shift detected in the growth rate in 1972, outlying observations are detected through two subsequent step-indicators with opposite-signs (e.g. in 1998/1999), as well as a downward step-shift at the end of the sample in 2004. This example demonstrates the flexibility of the SIS approach – step-shifts are easily identified even at the end of the sample while outliers can be detected simultaneously. The model can easily be extended to include autoregressive terms using the `ar` argument, for example we could estimate an AR(1) model with step-indicator saturation writing `isat(yso2, ar=1, t.pval=0.01)`.

6.2. Testing and bias correcting post-model selection in indicator saturation

The coefficient path of the intercept of the `isat` object can be extracted using the `isatvar` function. The function returns the coefficient path both as the time-varying intercept (`const.path`) and as deviation relative to the full-sample intercept (`coef.path`), together with the approx-

Figure 1: Annual UK SO_2 emission growth rate: `isat` Model results. Top panel shows observed (blue) and fit (red). Middle panel shows the standardised residuals, bottom panel shows the coefficient path relative to the intercept and its approximate 95% confidence interval.



imate variance of the coefficient path computed using the approach in Pretis (2015). When the model is specified to include autoregressive terms, then `isatvar` (setting `lr=TRUE`) also returns the static long-run solution of the dynamic model with its approximate variance.[7]

```
sisvar <- isatvar(sis)

       coef.path  const.path    const.var    const.se
1946  0.00000000  0.01465385 6.291637e-05 0.007931984
1947  0.00000000  0.01465385 6.291637e-05 0.007931984
...
```

The terminal models of `isat` are the result of model selection, and may therefore lead to a selection bias in the estimates. Post-selection bias-correction for orthogonal variables can be conducted using the method proposed in Hendry and Krolzig (2005). This is implemented as the function `biascorr`. Following Pretis (2015), bias-correction of the coefficients in a SIS model can be directly applied to the coefficient path without prior orthogonalisation. Bias-correcting the coefficient path of the above model of the growth rate of SO_2 yields the one- and two-step bias-corrected coefficients:

```
bcorr <- biascorr(b=sisvar[,"const.path"], b.se=sisvar[,"const.se"],
       p.alpha=0.01, T=length(sisvar[,"const.path"]))
```

[7] Note that the consistency correction for the estimate of the residual variance when using IIS in Johansen and Nielsen (2016) is not implemented, thus the variance results in `isatvar` are approximate.

```
         beta    beta.1step   beta.2step
...
1997 -0.14560000 -0.14560000 -0.14560000
1998 -0.01700000 -0.01700000 -0.01700000
1999 -0.30100000 -0.30099983 -0.30099983
2000 -0.05550000 -0.04043232 -0.03000334
2001 -0.05550000 -0.04043232 -0.03000334
...
```

The function `isattest` makes it possible to conduct hypothesis tests on the coefficient path of the intercept of an `isat` object. This test is described in Pretis (2015) and builds on Ericsson (2016) and Pretis *et al.* (2015) who use indicator saturation as a test for time-varying forecast accuracy. The main arguments of the `isattest` function are:

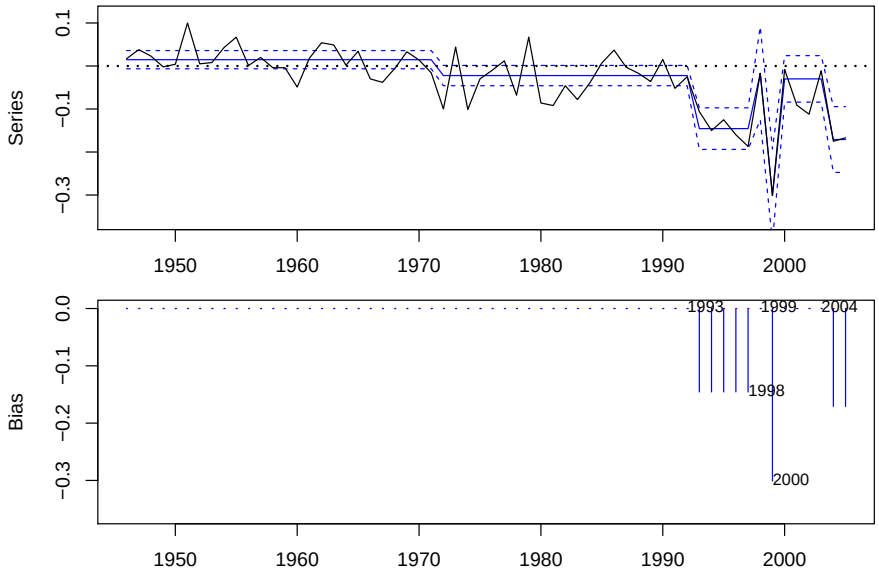
- `hnull`: numeric. The null-hypothesis value to be tested against.
- `lr`: logical. If `TRUE` and the `isat` object to be tested contains autoregressive terms, then the test is conducted on the long-run equilibrium coefficient path.
- `ci.pval`: numeric, between 0 and 1. The level of significance for the confidence interval and hypothesis test.
- `biascorr`: logical. If `TRUE` then the coefficient path is bias-corrected prior to testing. This is only valid for a non-dynamic (no auto-regressive terms) test without additional covariates.

Here we test the time-varying mean (as determined using SIS) of the annual growth rate of UK SO_2 emissions against the null hypothesis of zero-growth using `isattest`:

```
isattest(sis, hnull=0, lr=FALSE, ci.pval = 0.99, plot.turn = TRUE,
        biascorr=TRUE)

           ci.low      ci.high bias.high   bias.low
1946 -0.006539007  0.035846700         0  0.0000000
1947 -0.006539007  0.035846700         0  0.0000000
1948 -0.006539007  0.035846700         0  0.0000000
...
```

The results are shown in the automatically-generated plot given in Figure 2 (the `plot.turn=TRUE` argument automatically adds the break dates into the plot in the lower panel). When testing at 1% and using bias-correction this suggests that the detected shift in 1972 does not significantly move the growth-rate away from zero. Similarly, the upward shift in 2000 moves the growth rate back to zero. This change, however, is off-set by the shift at the end of the sample which shows the growth rate turning significantly negative in 2004.

Figure 2: Hypothesis test on the annual UK SO_2 emission growth rate using `isattest`. Top panel shows observed (black) and bias-corrected (`biascorr=TRUE`) fit (blue). Bottom panel shows the periods where the null-hypothesis is rejected, together with the dates of the significant breaks (`plot.turn=TRUE`).



7. Generating LaTeX code

The objects returned by `arx`, `getsm`, `getsv` and `isat` are lists. The entries in these lists containing the main estimation output are objects of class `data.frame`. That means we can use the R package **xtable** to generate LaTeXcode of these tables. For example, in Section 4.2 we stored the estimation results of an AR(2)-X-log-ARCH(3)-X model in `mod05`. This is a list of class `arx`. The entries of the list that contain data frames with the estimation and diagnostics results are `mean.results`, `variance.results` and `diagnostics`. The following prints the LaTeXcode associated with the table `mean.results`:

```
library(xtable)
print( xtable(mod05$mean.results) )
```

The printed LaTeXcode is:

```
% latex table generated in R 3.2.2 by xtable 1.8-0 package
% Fri Feb 26 17:16:36 2016
\begin{table}[ht]
\centering
\begin{tabular}{rrrrr}
\hline
& coef & std.error & t-stat & p-value \\
\hline
mconst & -0.06 & 0.08 & -0.76 & 0.45 \\
```

```
ar1 & 0.19 & 0.12 & 1.57 & 0.12 \\
ar2 & 0.03 & 0.11 & 0.30 & 0.76 \\
mxreg1 & 0.12 & 0.08 & 1.45 & 0.15 \\
mxreg2 & 0.01 & 0.09 & 0.13 & 0.89 \\
mxreg3 & -0.11 & 0.08 & -1.33 & 0.19 \\
mxreg4 & -0.22 & 0.10 & -2.18 & 0.03 \\
mxreg5 & 0.00 & 0.07 & 0.02 & 0.99 \\
\hline
\end{tabular}
\end{table}
```

Similarly, `print(xtable(mod05$variance.results))` prints the LaTeXcode associated with the log-variance specification, and `print(xtable(mod05$diagnostics))` prints the LaTeXcode associated with the diagnostics. See the **xtable** documentation for further options.

8. Exporting results to EViews and STATA

The two most popular commercial econometric softwares are **EViews** and **STATA**, but none of these provide GETS modelling capabilities. To facilitate the usage of GETS modelling for **EViews** and **STATA** users, we provide two functions for this purpose, `eviews` and `stata`. Both functions work in a similar way, and both can be applied on either `arx`, `gets` or `isat` objects. For example, typing `eviews(getsm05)` yields the following print:

```
EViews code to estimate the model:

  equation getsm05.ls(cov=white) yy mxreg4

R code (example) to export the data of the model:

  eviews(getsm05, file='C:/Users/myname/Documents/getsdata.csv')
```

In other words, the code to estimate the final model in **EViews**, and – if needed – a code-suggestion for how to export the data of the model. The need to export the data of the final model is likely to be most relevant subsequent to the use of `isat`. The `stata` function works similarly. Note that both the `eviews` and `stata` functions are only applicable to conditional mean specifications, since neither EViews nor **STATA** offer the estimation of dynamic log-variance models.

References

Akaike H (1974). "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control*, **19**, 716–723.

Campos J, Hendry DF, Ericsson NR (eds.) (2005). *General-to-Specific Modeling. Volumes 1 and 2*. Edward Elgar Publishing, Cheltenham.

Castle JL, Doornik JA, Hendry DF, Pretis F (2015). "Detecting location shifts during model selection by step-indicator saturation." *Econometrics*, **3**(2), 240–264.

Doornik J (2009). "Autometrics." In JL Castle, N Shephard (eds.), *The Methodology and Practice of Econometrics: A Festschrift in Honour of David F. Hendry*, pp. 88–121. Oxford University Press, Oxford.

Doornik JA, Hendry DF (2007). *Empirical Econometric Modelling - PcGive 12: Volume I*. Timberlake Consultants Ltd., London.

Duan N (1983). "Smearing Estimate: A Nonparametric Retransformation Method." *Journal of the Americal Statistical Association*, **78**, pp. 605–610.

Engle R (1982). "Autoregressive Conditional Heteroscedasticity with Estimates of the Variance of United Kingdom Inflations." *Econometrica*, **50**, 987–1008.

Ericsson NR (2016). "How Biased Are US Government Forecasts of the Federal Debt?" *International Journal of Forecasting*.

Glosten LR, Jagannathan R, Runkle DE (1993). "On the Relation between the Expected Value and the Volatility of the Nominal Excess Return on Stocks." *Journal of Finance*, **48**, 1779–1801.

Hannan E, Quinn B (1979). "The Determination of the Order of an Autoregression." *Journal of the Royal Statistical Society. Series B*, **41**, 190–195.

Hendry DF (2003). "J. Denis Sargan and the Origins of LSE Econometric Methodology." *Econometric Theory*, **19**, 457–480.

Hendry DF, Doornik J (2014). *Empirical Model Discovery and Theory Evaluation*. The MIT Press, London.

Hendry DF, Johansen S (2015). "Model discovery and Trygve HaavelmoâĂŹs legacy." *Econometric Theory*, **31**(01), 93–114.

Hendry DF, Johansen S, Santos C (2007). "Automatic selection of indicators in a fully saturated regression." *Computational Statistics*, **20**, 3–33. DOI 10.1007/s00180-007-0054-z.

Hendry DF, Krolzig HM (2001). *Automatic Econometric Model Selection using PcGets*. Timberlake Consultants Press, London.

Hendry DF, Krolzig HM (2005). "The Properties of Automatic Gets Modelling." *Economic Journal*, **115**, C32–C61.

Hoover KD, Perez SJ (1999). "Data Mining Reconsidered: Encompassing and the General-to-Specific Approach to Specification Search." *Econometrics Journal*, **2**, 167–191. Dataset and code: http://www.csus.edu/indiv/p/perezs/Data/data.htm.

Jarque C, Bera A (1980). "Efficient Tests for Normality, Homoskedasticity, and Serial Independence." *Economics Letters*, **6**, pp. 255–259.

Johansen S, Nielsen B (2016). "Asymptotic theory of outlier detection algorithms for linear time series regression models." *Scandinavian Journal of Statistics*.

Ljung G, Box G (1979). "On a Measure of Lack of Fit in Time Series Models." *Biometrika*, **66**, 265–270.

Lovell MC (1983). "Data Mining." *The Review of Economics and Statistics*, **65**, 1–12.

Mizon G (1995). "Progressive Modeling of Macroeconomic Time Series: The LSE Methodology." In KD Hoover (ed.), *Macroeconometrics. Developments, Tensions and Prospects*, pp. 107–169. Kluwer Academic Publishers.

Newey W, West K (1987). "A Simple Positive Semi-Definite, Heteroskedasticity and Autocorrelation Consistent Covariance Matrix." *Econometrica*, **55**, 703–708.

Pretis F (2015). "Testing for time-varying predictive accuracy: a bias-corrected dynamic indicator saturation approach." *University of Oxford Economics Discussion Paper*.

Pretis F, Mann ML, Kaufmann RK (2015). "Testing competing models of the temperature hiatus: assessing the effects of conditioning variables and temporal uncertainties through sample-wide break detection." *Climatic Change*, **131**(4), 705–718.

Pretis F, Schneider L, Smerdon JE, Hendry DF (2016). "Detecting Volcanic Eruptions in Temperature Reconstructions by Designed Break-Indicator Saturation." *Journal of Economic Surveys*, (forthcoming).

Ryan JA, Ulrich JM (2014). *xts: eXtensible Time Series*. R package version 0.9-7, URL https://CRAN.R-project.org/package=xts.

Schwarz G (1978). "Estimating the Dimension of a Model." *The Annals of Statistics*, **6**, 461–464.

Smith SJ, Aardenne Jv, Klimont Z, Andres RJ, Volke A, Delgado Arias S (2011). "Anthropogenic sulfur dioxide emissions: 1850–2005." *Atmospheric Chemistry and Physics*, **11**(3), 1101–1116.

Sucarrat G (2010). "Econometric Reduction Theory and Philosophy." *The Journal of Economic Methodology*, **17**, 53–75.

Sucarrat G (2014). *lgarch: Simulation and estimation of log-GARCH models*. http://cran.r-project.org/web/packages/lgarch/.

Sucarrat G, Escribano Á (2012). "Automated Model Selection in Finance: General-to-Specific Modelling of the Mean and Volatility Specifications." *Oxford Bulletin of Economics and Statistics*, **74**, 716–735.

Sucarrat G, Grønneberg S, Escribano Á (2015). "Estimation and Inference in Univariate and Multivariate Log-GARCH-X Models When the Conditional Density is Unknown." *Computational Statistics and Data Analysis*. Forthcoming. DOI: http://dx.doi.org/10.1016/j.csda.2015.12.005. Working Paper version: http://mpra.ub.uni-muenchen.de/62352/.

White H (1980). "A Heteroskedasticity-Consistent Covariance Matrix and a Direct Test for Heteroskedasticity." *Econometrica*, **48**, 817–838.

Zeileis A, Grothendieck G (2005). "zoo: S3 Infrastructure for Regular and Irregular Time Series." *Journal of Statistical Software*, **14**(6), 1–27. URL http://www.jstatsoft.org/v14/i06/.

Affiliation:

Felix Pretis
Programme for Economic Modelling at the Oxford Martin School
& Department of Economics
University of Oxford
E-mail: felix.pretis@nuffield.ox.ac.uk
URL: http://www.felixpretis.org/

James Reade
Programme for Economic Modelling at the Oxford Martin School
& Department of Economics University of Reading
E-mail: j.j.reade@reading.ac.uk
URL: https://sites.google.com/site/jjamesreade/

Genaro Sucarrat
Department of Economics
BI Norwegian Business School
0484 Oslo, Norway
E-mail: genaro.sucarrat@bi.no
URL: http://www.sucarrat.net/