Using the vars package

Dr. Bernhard Pfaff, Kronberg im Taunus

July 24, 2007

Abstract

The utilisation of the functions contained in the package 'vars' are explained by employing a data set of the Canadian economy. The package's scope includes functions for estimating vector autoregressive (henceforth: VAR) and structural vector autoregressive models (henceforth: SVAR). In addition to the two cornerstone functions `VAR()` and `SVAR()` for estimating such models, functions for diagnostic testing, estimation of restricted VARs, prediction of VARs, causality analysis, impulse response analysis (henceforth: IRA) and forecast error variance decomposition (henceforth: FEVD) are provided too. In each section, the underlying concept and/or method is briefly explained, thereby drawing on the exibitions in Lütkepohl [2006], Lütkepohl, Krätzig, Phillips, Ghysels & Smith [2004], Lütkepohl & Breitung [1997], Hamilton [1994] and Watson [1994].

The package's code is purely written in `R` and S3-classes with methods have been utilised. It is shipped with a NAMESPACE and a ChangeLog file. It has dependencies to `MASS` (see Venables & Ripley [2002]) and `strucchange` (see Zeileis, Leisch, Hornik & Kleiber [2002]).

Contents

1 Introduction **2**

2 VAR: Vector autoregressive models **2**
2.1 Definition . . . 2
2.2 Estimation . . . 3
2.3 Restricted VARs . . . 6
2.4 Diagnostic testing . . . 9
2.5 Causality Analysis . . . 15
2.6 Forecasting . . . 18
2.7 Impulse response analysis . . . 20
2.8 Forecast error variance decomposition . . . 22

3 SVAR: Structural vector autoregressive models **24**
3.1 Definition . . . 24
3.2 Estimation . . . 25
3.3 Impulse response analysis . . . 29
3.4 Forecast error variance decomposition . . . 29

4 VECM to VAR **29**

1 Introduction

Since the critique of Sims [1980] in the early eighties of the last century, VAR analysis has evolved as a standard instrument in econometrics for analysing multivariate time series. Because statistical tests are highly used in determining interdependencies and dynamic relationships between variables, it became soon evident to enrich this methodology by incorporating non-statistical *a priori* information, hence SVAR models evolved which try to bypass these shortcomings. At the same time as Sims jeopardised the paradigm of multiple structural equation models laid out by the Cowles Foundation in the fourties and fifties of the last century, Granger [1981] and Engle & Granger [1987] endowed econometricians with a powerful tool for modelling and testing economic relationships, namely, the concept of integration and and cointegration. Nowadays these traces of research are unified in the form of vector error correction and structural vector error correction models. Although these latter topics are left out in this vignette, the interested reader is referred to the monographies of Lütkepohl [2006], Hendry [1995], Johansen [1995], Hamilton [1994], Banerjee, Dolado, Galbraith & Hendry [1993] and Pfaff [2006] for an exhibition of unit root tests and co-integration analysis in `R`.

To the author's knowledge, currently only functions in the base distribution of `R` and in the `CRAN`-packages `dse` (package bundle, see Gilbert 1993, 1995, 2000) and `fMultivar` (Würtz 2006) are made available for estimating ARIMA and VARIMA time series models. Though, the `CRAN`-package `MSBVAR` (Brandt 2006) provides methods for estimating frequentist and Bayesian Vector Autoregression (BVAR) models, the methods and functions provided in the package `vars` try to fill a gap in the econometrics' methods landscape of `R` by providing the 'standard' tools in the context of VAR and SVAR analysis.

The vignette is structured as follows: the next section is entirely devoted to VARs (definition, estimation, restrictions, diagnostic testing, forecasting, IRA and FEVD). In the ensuing section the topic of SVAR is elucidated (definition, estimation, IRA and FEVD). Finally, the transformation of a VECM to a VAR in levels is exhibited with its associated methods and functions.

2 VAR: Vector autoregressive models

2.1 Definition

In its basic form, a VAR consists of a set of K endogenous variables $\mathbf{y}_t = (y_{1t}, \ldots, y_{kt}, \ldots, y_{Kt})$ for $k = 1, \ldots K$. The VAR(p)-process is then defined as:

$$\mathbf{y}_t = A_1\mathbf{y}_{t-1} + \ldots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t \quad , \tag{1}$$

with A_i are $(K \times K)$ coefficient matrices for $i = 1, \ldots, p$ and $\mathbf{u}_t$ is a K-dimensional white noise process with time invariant positive definite covariance matrix $E(\mathbf{u}_t\mathbf{u}_t') = \Sigma_{\mathbf{u}}$.[1]

par One important characteristic of a VAR(p)-process is its stability. This

[1] Vectors are assigned by small bold letters and matrices by capital letters. Scalars are written out as small letters. which are possibly subscripted.

means that it generates stationary time series with time invariant means, variances and covariance structure, given sufficient starting values. One can check this by evaluating the reverse characteristic polynomial:

$$\det(I_K - A_1 z - \ldots - A_p z^p) \neq 0 \quad \text{for } |z| \leq 1 \; . \tag{2}$$

If the solution of the above equation has a root for $z = 1$, then either some or all variables in the VAR(p)-process are integrated of order one, *i.e.* $I(1)$. It might be the case, that cointegration between the variables does exist. This instance can be better analysed in the context of a vector-error-correction model (VECM). The reader is referred to the monography of Johansen [1995] for a theoretical exposition and to Pfaff [2006] for an implementation with the `CRAN`-package 'urca' in `R`.

In practice, the stability of an empirical VAR(p)-process can be analysed by considering the companion form and caculating the *eigenvalues* of the coefficient matrix. A VAR(p)-process can be written as a VAR(1)-process as:

$$\xi_t = A\xi_{t-1} + \mathbf{v}_t \; , \tag{3}$$

with:

$$\xi_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \end{bmatrix} \; , \; A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix} \; , \; \mathbf{v}_t = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix} \; , \tag{4}$$

whereby the dimensions of the stacked vectors ξ_t and $\mathbf{v}_t$ is $(Kp \times 1)$ and of the matrix A is $(Kp \times Kp)$. If the moduli of the *eigenvalues* of A are less than one, then the VAR(p)-process is stable. The calculation of the *eigenvalues* is made available with the function `roots()`. The function has an argument `'modulus'` of type logical that returns by default the moduli of the *eigenvalues*, otherwise a vector of complex numbers is returned (for an application see section 2.2 below).

2.2 Estimation

For a given sample of the endogenous variables $\mathbf{y}_1, \ldots \mathbf{y}_T$ and sufficient presample values $\mathbf{y}_{-p+1}, \ldots, \mathbf{y}_0$, the coefficients of a VAR(p)-process can be estimated efficiently by least-squares applied separately to each of the equations.

Before considering the implementation in the package 'vars', let us briefly discuss the utilised data set and plot the series (see figure 1). The original time series are published by the OECD and the transformed series, as described in the help page to `Canada` are provided with `JMulti` (see Lütkepohl et al. [2004]). The sample range is from the 1stQ 1980 until 4thQ 2000.

```
> library(vars)
> data(Canada)
> layout(matrix(1:4, nrow = 2, ncol = 2))
> plot.ts(Canada$e, main = "Employment", ylab = "", xlab = "")
> plot.ts(Canada$prod, main = "Productivity", ylab = "", xlab = "")
> plot.ts(Canada$rw, main = "Real Wage", ylab = "", xlab = "")
> plot.ts(Canada$U, main = "Unemployment Rate", ylab = "", xlab = "")
```

Figure 1: Canada: Macroeconomic series

The variable `e` is used for employment; `prod` is a measure of labour productivity; `rw` assigns the real wage and finally `U` is the unemployment rate. The function for estimating a VAR is `VAR()`. It consists of three arguments: the data matrix object `y` (or an object that can be coerced to a `matrix`), the integer lag-order `p` and the type of deterministic regressors to include into the VAR(p). An optimal lag-order can be determined according to an information criteria or the final prediction error of a VAR(p) with the function `VARselect()`. Its arguments are exhibited in the code snippet below.

```
> args(VAR)

function (y, p = 1, type = c("const", "trend", "both", "none"),
    season = NULL, exogen = NULL)
NULL

> args(VARselect)

function (y, lag.max = 10, type = c("const", "trend", "both",
    "none"), season = NULL, exogen = NULL)
NULL

> VARselect(Canada, lag.max = 5, type = "const")

$selection
AIC(n)  HQ(n)  SC(n) FPE(n)
     3      2      2      3
```

```
$criteria
                  1           2           3           4           5
AIC(n) -5.817851996 -6.35093701 -6.397756084 -6.145942174 -5.926500201
HQ(n)  -5.577529641 -5.91835677 -5.772917961 -5.328846166 -4.917146309
SC(n)  -5.217991781 -5.27118862 -4.838119523 -4.106417440 -3.407087295
FPE(n)  0.002976003  0.00175206  0.001685528  0.002201523  0.002811116
```

The `VARselect()` enables the user to determine an optimal lag length according to an information criteria or the final prediction error of an empirical VAR(p)-process. Each of these measures are defined in the function's help file. The function returns a list object with the optimal lag-order according to each of the criteria, as well as a matrix containing the values for all lags up to `lag.max`. According to the more conservative $SC(n)$ and $HQ(n)$ criteria, the empirical optimal lag-order is 2. Please note, that the calculation of these criteria is based upon the same sample size, and hence the criteria might take slightly different values whence a VAR for the choosen order is estimated.

In a next step, the VAR(2) is estimated with the function `VAR()` and as deterministic regressors a constant is included.

```
> var.2c <- VAR(Canada, p = 2, type = "const")
> names(var.2c)
```

```
[1] "varresult"    "datamat"      "y"            "type"         "p"
[6] "K"            "obs"          "totobs"       "restrictions" "call"
```

The function returns a list object of class `varest` and the list elements are explained in detail in the function's help file. Let us now focus on two methods, namely `summary` and `plot`.

```
> summary(var.2c)
> plot(var.2c)
```

The `summary` method simply applies the `summary.lm` method to the `lm` objects contained in `varresult`. The OLS results of the example are shown in separate tables 1 – 4 below. It turns out, that not all lagged endogenous variables enter significanctly into the equations of the VAR(2). The estimation of a restricted VAR is the topic of the next section.

	Estimate	Std. Error	t value	Pr(>|t|)
e.l1	1.6378	0.1500	10.92	0.0000
prod.l1	0.1673	0.0611	2.74	0.0078
rw.l1	−0.0631	0.0552	−1.14	0.2569
U.l1	0.2656	0.2028	1.31	0.1944
e.l2	−0.4971	0.1595	−3.12	0.0026
prod.l2	−0.1017	0.0661	−1.54	0.1282
rw.l2	0.0038	0.0555	0.07	0.9450
U.l2	0.1327	0.2073	0.64	0.5242
const	−136.9984	55.8481	−2.45	0.0166

Table 1: Regression result for employment equation

Before we proceed with the estimation of restricted VARs, let us first look at the `plot` method for objects with class attribute `varest` and the

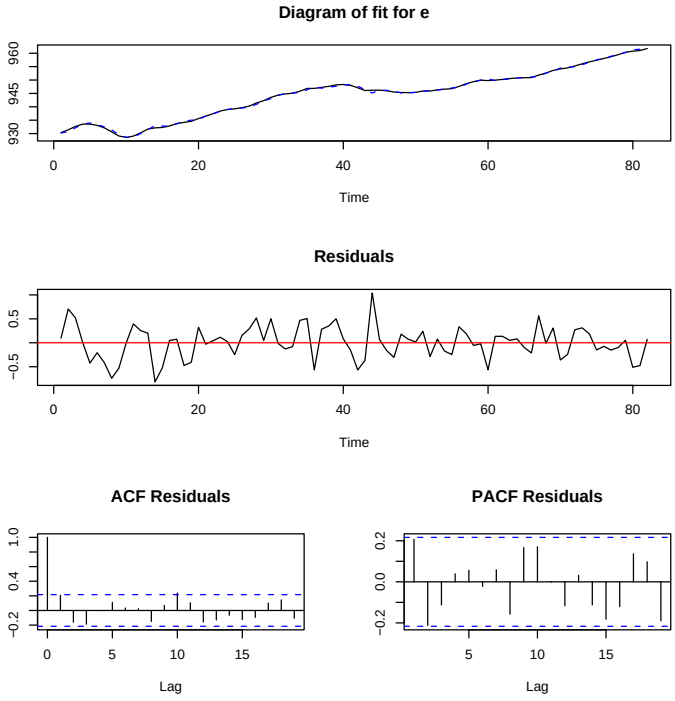

Figure 2: Employment equation: Diagram of fit, residuals with ACF and PACF

`roots()` function for checking the VAR's stability, that was briefly mentioned at the end of the previous section. For each equation in a VAR, a plot consisting of a diagram of fit, a residual plot, the autocorrelation and partial autocorrelation function of the residuals are shown. If the plot method is called interactively, the user is requested to enter `<RETURN>` for commencing to the next plot. Currently, the ...-argument of the `plot` method is unused. However, given the information contained in an object of class `varest`, it is fairly easy to set up such plots and tailormade them to her/his needs. The plots of the four equations are shown in the exhibits 2 to 5. Whence, we have estimated a VAR(p), we should check its stability. Here, stability does not refer to the coefficients' stability, *i.e.* the stability of the regressions, but rather the stability of the system of difference equations. As pointed out above, if the moduli of the *eigenvalues* of the companion matrix are less than one, the system is stable.

```
> roots(var.2c)

[1] 0.9950338 0.9081062 0.9081062 0.7380565 0.7380565 0.1856381 0.1428889
[8] 0.1428889
```

Although, the first *eigenvalue* is pretty close to unity, for the sake of simplicity, we assume a stable VAR(2)-process with a constant as deterministic regressor.

	Estimate	Std. Error	t value	Pr(>|t|)
e.l1	−0.1728	0.2698	−0.64	0.5239
prod.l1	1.1504	0.1099	10.46	0.0000
rw.l1	0.0513	0.0993	0.52	0.6071
U.l1	−0.4785	0.3647	−1.31	0.1936
e.l2	0.3853	0.2869	1.34	0.1835
prod.l2	−0.1724	0.1188	−1.45	0.1510
rw.l2	−0.1189	0.0998	−1.19	0.2378
U.l2	1.0159	0.3728	2.72	0.0080
const	−166.7755	100.4339	−1.66	0.1011

Table 2: Regression result for productivity equation

	Estimate	Std. Error	t value	Pr(>|t|)
e.l1	−0.2688	0.3226	−0.83	0.4074
prod.l1	−0.0811	0.1315	−0.62	0.5395
rw.l1	0.8955	0.1188	7.54	0.0000
U.l1	0.0121	0.4361	0.03	0.9779
e.l2	0.3678	0.3431	1.07	0.2872
prod.l2	−0.0052	0.1421	−0.04	0.9710
rw.l2	0.0527	0.1194	0.44	0.6604
U.l2	−0.1277	0.4459	−0.29	0.7754
const	−33.1883	120.1105	−0.28	0.7831

Table 3: Regression result for real wage equation

2.3 Restricted VARs

From tables 1-4 it is obvious that not all regressors enter significanctly. With the function `restrict()` the user has the option to re-estimate the VAR either by significance (argument `method = 'ser'`) or by imposing zero restrictions manually (argument `method = 'manual'`). In the former case, each equation is re-estimated separately as long as there are t-values that are in absolut value below the threshhold value set by the function's argument `thresh`. In the latter case, a restriction matrix has to be provided that consists of 0/1 values, thereby selecting the coefficients to be retained in the model. The function's arguments are therefore:

```
> args(restrict)

function (x, method = c("ser", "manual"), thresh = 2, resmat = NULL)
NULL

> var2c.ser <- restrict(var.2c, method = "ser", thresh = 2)
> var2c.ser$restrictions

     e.l1 prod.l1 rw.l1 U.l1 e.l2 prod.l2 rw.l2 U.l2 const
e       1       1     1    1    1       0     0    0     1
prod    0       1     0    0    1       0     1    1     1
rw      0       1     1    0    1       0     0    1     0
U       1       0     0    1    1       0     1    0     1

> B(var2c.ser)
```

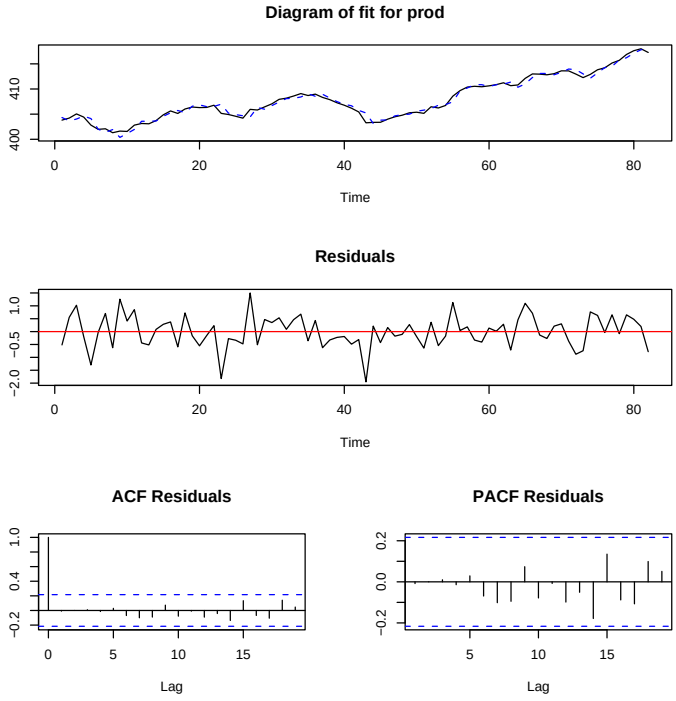

Figure 3: Productivity equation: Diagram of fit, residuals with ACF and PACF

```
           e.l1     prod.l1       rw.l1      U.l1        e.l2 prod.l2
e     1.7245893  0.07872263 -0.05370603 0.3915061 -0.60070476       0
prod  0.0000000  1.00809918  0.00000000 0.0000000  0.28943990       0
rw    0.0000000 -0.11816412  0.96382332 0.0000000  0.07092345       0
U    -0.6954549  0.00000000  0.00000000 0.5600228  0.52253799       0
           rw.l2       U.l2     const
e     0.00000000  0.0000000 -128.8945
prod -0.09728839  0.7503647 -240.5605
rw    0.00000000 -0.1930013    0.0000
U     0.05670873  0.0000000  142.6316
```

```
> res <- matrix(rep(1, 36), nrow = 4, ncol = 9)
> res[1, 3] <- 0
> res[1, 4] <- 0
> var2c.man <- restrict(var.2c, method = "manual", resmat = res)
> var2c.man$restrictions
```

```
     e.l1 prod.l1 rw.l1 U.l1 e.l2 prod.l2 rw.l2 U.l2 const
e       1       1     0    0    1       1     1    1     1
prod    1       1     1    1    1       1     1    1     1
rw      1       1     1    1    1       1     1    1     1
U       1       1     1    1    1       1     1    1     1
```

```
> B(var2c.man)
```

```
           e.l1     prod.l1      rw.l1        U.l1       e.l2       prod.l2
e     1.5079689  0.16584386 0.00000000  0.00000000 -0.4043821 -0.095383974
```

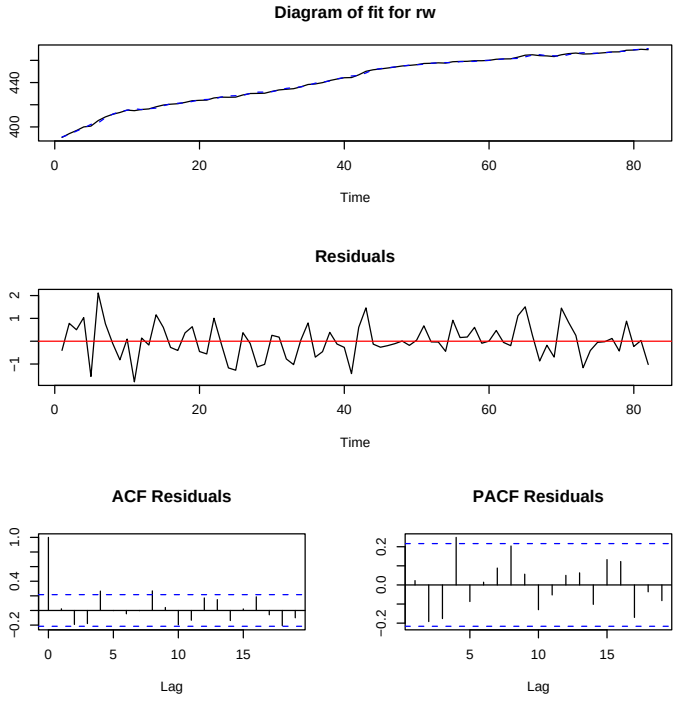

Figure 4: Real wage equation: Diagram of fit, residuals with ACF and PACF

```
prod -0.1727658  1.15042820 0.05130390 -0.47850131  0.3852589 -0.172411873
rw   -0.2688329 -0.08106500 0.89547833  0.01213003  0.3678489 -0.005180947
U    -0.5807638 -0.07811707 0.01866214  0.61893150  0.4098182  0.052116684
           rw.l2        U.l2       const
e    -0.04593897  0.33088442 -109.20640
prod -0.11885104  1.01591801 -166.77552
rw    0.05267656 -0.12770826  -33.18834
U     0.04180115 -0.07116885  149.78056
```

In the example above, the third and fourth coefficient of the employment equation (*i.e.*, the first equation) are set to zero for `method = 'manual'`. The function `restrict()` returns a list object with class attribute `varest`. The coefficients of these objects can be displayed conveniently by either `B()` (all coeffiecients, including deterministic coefficients) or by `A()` (only coefficients of lagged endogenous variables). The output of the former is shown in the code snippet above. It should be noted at this point, that a restricted VAR is estimated by OLS too, instead of employing the EGLS method (see Lütkepohl [2006] for an exposition and section ??).

2.4 Diagnostic testing

In package 'vars' the functions for diagnostic testing are `arch()`, `normality()`, `serial()` and `stability()`. The former three functions return a list object with class attribute `varcheck` for which a `plot`-method exists. The plots, one for each equation, include a residual plot, an empirical distribution plot and the ACFs and PACFs of the residuals and their

	Estimate	Std. Error	t value	Pr(>|t|)
e.l1	−0.5808	0.1156	−5.02	0.0000
prod.l1	−0.0781	0.0471	−1.66	0.1017
rw.l1	0.0187	0.0426	0.44	0.6625
U.l1	0.6189	0.1563	3.96	0.0002
e.l2	0.4098	0.1230	3.33	0.0014
prod.l2	0.0521	0.0509	1.02	0.3095
rw.l2	0.0418	0.0428	0.98	0.3319
U.l2	−0.0712	0.1598	−0.45	0.6574
const	149.7806	43.0481	3.48	0.0009

Table 4: Regression result for unemployment equation

squares. The function `stability()` returns a list object with class attribute `varstabil`. The function itself is just a wrapper for the function `efp()` from package `strucchange`. The first element of the returned list object is itself a list of objects with class attribute `efp`. Hence, the `plot`-method for objects of class `varstabil` just call the `plot`-method for objects of class `efp`. Let us now turn to each of these functions in more detail.

The implemented tests for heteroscedasticity are the univariate and multivariate ARCH test (see Engle [1982], Hamilton [1994] and Lütkepohl [2006]). The multivariate ARCH-LM test is based on the following regression (the univariate test can be considered as special case of the exhibtion below and is skipped):

$$vech(\hat{\mathbf{u}}_t\hat{\mathbf{u}}_t{}') = \beta_0 + B_1 vech(\hat{\mathbf{u}}_{t-1}\hat{\mathbf{u}}'_{t-1}) + \ldots + B_q vech(\hat{\mathbf{u}}_{t-q}\hat{\mathbf{u}}'_{t-q}) + \mathbf{v}_t \quad , \tag{5}$$

whereby $\mathbf{v}_t$ assigns a spherical error process and *vech* is the column-stacking operator for symmetric matrices that stacks the columns from the main diagonal on downwards. The dimension of β_0 is $\frac{1}{2}K(K+1)$ and for the coefficient matrices B_i with $i = 1, \ldots, q$, $\frac{1}{2}K(K+1) \times \frac{1}{2}K(K+1)$. The null hypothesis is: $H_0 := B_1 = B_2 = \ldots = B_q = 0$ and the alternative is: $H_1 : B_1 \neq 0 \cap B_2 \neq 0 \cap \ldots \cap B_q \neq 0$. The test statistic is defined as:

$$VARCH_{LM}(q) = \frac{1}{2}TK(K+1)R_m^2 \quad , \tag{6}$$

with

$$R_m^2 = 1 - \frac{2}{K(K+1)} tr(\hat{\Omega}\hat{\Omega}_0^{-1}) \quad , \tag{7}$$

and $\hat{\Omega}$ assigns the covariance matrix of the above defined regression model. This test statistic is distributed as $\chi^2(qK^2(K+1)^2/4)$.
In the code example below this test is applied to the `var.2c` object.

```
> args(arch)

function (x, lags.single = 16, lags.multi = 5)
NULL

> var2c.arch <- arch(var.2c)
> names(var2c.arch)

[1] "resid"    "arch.uni" "arch.mul"

> var2c.arch
```

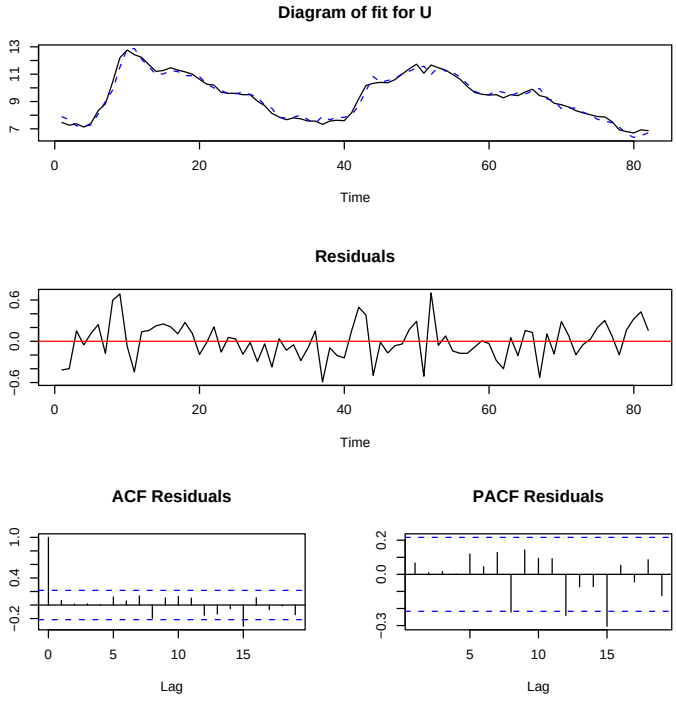

Figure 5: Unemployment equation: Diagram of fit, residuals with ACF and PACF

```
$e

        ARCH test (univariate)

data:  Residual of e equation
Chi-squared = 11.2611, df = 16, p-value = 0.7931


$prod

        ARCH test (univariate)

data:  Residual of prod equation
Chi-squared = 7.8221, df = 16, p-value = 0.954


$rw

        ARCH test (univariate)

data:  Residual of rw equation
Chi-squared = 5.1427, df = 16, p-value = 0.995
```

```
$U

        ARCH test (univariate)

data:  Residual of U equation
Chi-squared = 12.1486, df = 16, p-value = 0.7337



        ARCH (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 538.8897, df = 500, p-value = 0.1112
```

The Jarque-Bera normality tests for univariate and multivariate series are implemented and applied to the residuals of a VAR(p) as well as seperate tests for multivariate skewness and kurtosis (see Bera & Jarque [1980], [1981] and Jarque & Bera [1987] and Lütkepohl [2006]). The univariate versions of the Jarque-Bera test are applied to the residuals of each equation. A multivariate version of this test can be computed by using the residuals that are standardized by a Choleski decomposition of the variance-covariance matrix for the centered residuals. Please note, that in this case the test result is dependant upon the ordering of the variables.

```
> var2c.norm <- normality(var.2c)
> names(var2c.norm)

[1] "resid"  "jb.uni" "jb.mul"

> var2c.norm

$e

        JB-Test (univariate)

data:  Residual of e equation
Chi-squared = 0.1535, df = 2, p-value = 0.9261


$prod

        JB-Test (univariate)

data:  Residual of prod equation
Chi-squared = 4.2651, df = 2, p-value = 0.1185


$rw

        JB-Test (univariate)

data:  Residual of rw equation
Chi-squared = 0.3348, df = 2, p-value = 0.8459


$U
```

```
        JB-Test (univariate)

data:  Residual of U equation
Chi-squared = 0.5664, df = 2, p-value = 0.7534


$JB

        JB-Test (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 5.094, df = 8, p-value = 0.7475


$Skewness

        Skewness only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 1.7761, df = 4, p-value = 0.7769


$Kurtosis

        Kurtosis only (multivariate)

data:  Residuals of VAR object var.2c
Chi-squared = 3.3179, df = 4, p-value = 0.5061
```

```
> plot(var2c.norm)
```

The test statistics for the mutlivariate case are defined as:

$$JB_{mv} = s_3^2 + s_4^2 \,, \tag{8}$$

whereby s_3^2 and s_4^2 are computed according to:

$$s_3^2 = T\mathbf{b}_1'\mathbf{b}_1/6 \tag{9a}$$

$$s_4^2 = T(\mathbf{b}_2 - \mathbf{3}_K)'(\mathbf{b}_2 - \mathbf{3}_k)/24 \,, \tag{9b}$$

with $\mathbf{b}_1$ and $\mathbf{b}_2$ are the third and fourth non-central moment vectors of the standardized residuals $\hat{\mathbf{u}}_t^s = \tilde{P}^-(\hat{\mathbf{u}}_t - \bar{\hat{\mathbf{u}}}_t)$ and $\tilde{P}$ is a lower triangular matrix with positive diagonal such that $\tilde{P}\tilde{P}' = \tilde{\Sigma}_{\mathbf{u}}$, *i.e.*, the Choleski decomposition of the residual covariance matrix. The test statistic JB_{mv} is distributed as $\chi^2(2K)$ and the multivariate skewness, s_3^2, and kurtosis test, s_4^2 are distributed as $\chi^2(K)$. Likewise to ARCH, these tests are returned in the list elements `jb.uni` and `jb.mul`, which consist of objects with class attribute `htest`. Hence, the `print`-method for these objects is employed in `print.varcheck`. In the code example above, the null hypothesis of normality cannot be rejected.

For testing the lack of serial correlation in the residuals of a VAR(p), a Portmanteau test and the LM test proposed by Breusch & Godfrey are implemented in the function `serial()`. For both tests small sample modifications are calculated too, whereby the modification for the LM has been introduced by Edgerton & Shukur [1999]. Likewise, to the function

`normality()`, the test statistics are list elements of the returned object and have class attribute `htest`.
The Portmanteau statistic is defined as:

$$Q_h = T \sum_{j=1}^{h} tr(\hat{C}_j' \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) \ , \tag{10}$$

with $\hat{C}_i = \frac{1}{T} \Sigma_{t=i+1}^{T} \hat{\mathbf{u}}_t \hat{\mathbf{u}}_{t-i}'$. The test statistic has an approximate $\chi^2(K^2 h - n^*)$ distribution, and n^* is the number of coefficients excluding deterministic terms of a VAR(p). The limiting distribution is only valid for h tending to infinity at suitable rate with growing sample size. Hence, the trade-off is between a decent approximation to the χ^2 distribution and a loss in power of the test, when h is choosen too large. The small sample properties of the test statistic:

$$Q_h^* = T^2 \sum_{j=1}^{h} \frac{1}{T-j} tr(\hat{C}_j' \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) \tag{11}$$

may be better, and is available as the second entry in the list element `pt.mul`.

```
> var2c.serial <- serial(var.2c, lags.pt = 16, lags.bg = 5)
> names(var2c.serial)
[1] "resid"  "pt.mul" "LMh"    "LMFh"
> var2c.serial$pt.mul
$PT1

        Portmanteau Test (asymptotic)

data:  Residuals of VAR object var.2c
Chi-squared = 205.3538, df = 224, p-value = 0.8092


$PT2

        Portmanteau Test (adjusted)

data:  Residuals of VAR object var.2c
Chi-squared = 231.5907, df = 224, p-value = 0.3497
> plot(var2c.serial)
```

The Breusch-Godfrey LM-statistic (see Breusch 1978, Godfrey 1978) is based upon the following auxiliary regressions:

$$\hat{\mathbf{u}}_t = A_1 \mathbf{y}_{t-1} + \ldots + A_p \mathbf{y}_{t-p} + CD_t + B_1 \hat{\mathbf{u}}_{t-1} + \ldots + B_h \hat{\mathbf{u}}_{t-h} + \varepsilon_t \ . \tag{12}$$

The null hypothesis is: $H_0 : B_1 = \cdots = B_h = 0$ and correspondingly the alternative hypothesis is of the form $H_1 : \exists B_i \neq 0 \ for \ i = 1, 2, \ldots, h$. The test statistic is defined as:

$$LM_h = T(K - tr(\tilde{\Sigma}_R^{-1} \tilde{\Sigma}_e)) \quad , \tag{13}$$

where $\tilde{\Sigma}_R$ and $\tilde{\Sigma}_e$ assign the residual covariance matrix of the restricted and unrestricted model, respectively. The test statistic LM_h is distributed as $\chi^2(hK^2)$ and is returned by the function `serial()` as list element `LMh` with class attribute `htest`.

```
> var2c.serial$LMh

$LMh

        Breusch-Godfrey LM test

data:  Residuals of VAR object var.2c
Chi-squared = 92.6282, df = 80, p-value = 0.1581

> var2c.serial$LMFh

$LMFh

        Edgerton-Shukur F test

data:  Residuals of VAR object var.2c
F statistic = 1.1186, df1 = 80, df2 = 199, p-value = 0.2648
```

Edgerton & Shukur [1999] proposed a small sample correction, which is defined as:

$$LMF_h = \frac{1-(1-R_r^2)^{1/r}}{(1-R_r^2)^{1/r}} \frac{Nr-q}{Km}\,, \tag{14}$$

with $R_r^2 = 1 - |\tilde{\Sigma}_e|/|\tilde{\Sigma}_R|$, $r = ((K^2m^2-4)/(K^2+m^2-5))^{1/2}$, $q = 1/2Km - 1$ and $N = T - K - m - 1/2(K-m+1)$, whereby n is the number of regressors in the original system and $m = Kh$. The modified test statistic is distributed as $F(hK^2, int(Nr-q))$. This test statistic is returned by `serial()` as list element `LMFh` and has class attribute `htest`.

The stability of the regression relationships in a VAR(p) can be assessed with the function `stability()`. An empirical fluctuation process is estimated for each regression by passing the function's arguments to the `efp()`-function contained in the package `strucchange`. The function `stability()`returns a list object with class attribute `varstabil`. The first element, `stability`, is itself a list with objects of class `efp`. For more information about `efp()` see its help file and the cited literature therein.

```
> args(stability)

function (x, type = c("Rec-CUSUM", "OLS-CUSUM", "Rec-MOSUM",
    "OLS-MOSUM", "RE", "ME", "Score-CUSUM", "Score-MOSUM", "fluctuation"),
    h = 0.15, dynamic = FALSE, rescale = TRUE)
NULL

> var2c.stab <- stability(var.2c, type = "OLS-CUSUM")
> names(var2c.stab)

[1] "stability" "names"     "K"

> plot(var2c.stab)
```

In the R-code example above, an OLS-Cusum test is applied to the `varest` object `var.2c`. The graphical output is displayed in figures 6–9 on the following pages. The null hypothesis of a stable relationship cannot be rejected for neither regression in the VAR.

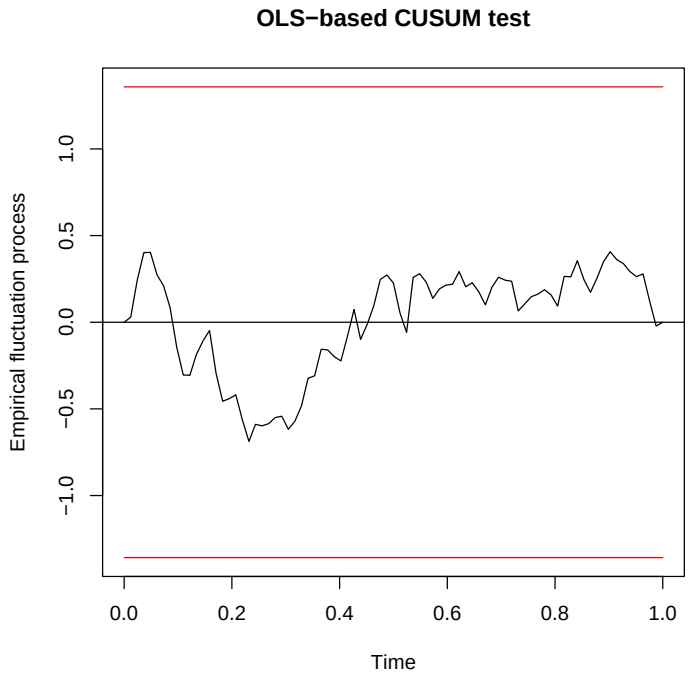

Figure 6: Employment equation: OLS-Cusum test

2.5 Causality Analysis

Often researchers are interested in the detection of causalities between variables. The most common one is the Granger-Causality test (Granger 1969). Incidentally, this test is not suited to test causal relationships in the strict sense, because the possibility of a *post hoc ergo propter hoc* fallacy cannot be excluded. This is true for any so called "causality test" in econometrics. It is therefore common practice to say that variables x does *granger-cause* variable y if variable x helps to predict variable y. Aside of this test, within the function `causality()` a Wald-type instantaneous causality test is implemented too. It is characterized by testing for nonzero correlation between the error processes of the cause and effect variables.

```
> args(causality)

function (x, cause = NULL)
NULL
```

The function `causality()` has two arguments. The first argument, `x`, is an object of class `varest` and the second, `cause`, is a character vector of the variable names, that are assumed to be causal to the remaining variables in a VAR(p)-process. If this argument is unset, then the variable in the first column of `x$y` is used as cause variable and a warning is printed.

For both tests the vector of endogenous variables $\mathbf{y}_t$ is split into two subvectors $\mathbf{y}_{1t}$ and $\mathbf{y}_{2t}$ with dimensions $(K_1 \times 1)$ and $(K_2 \times 1)$ with $K =$

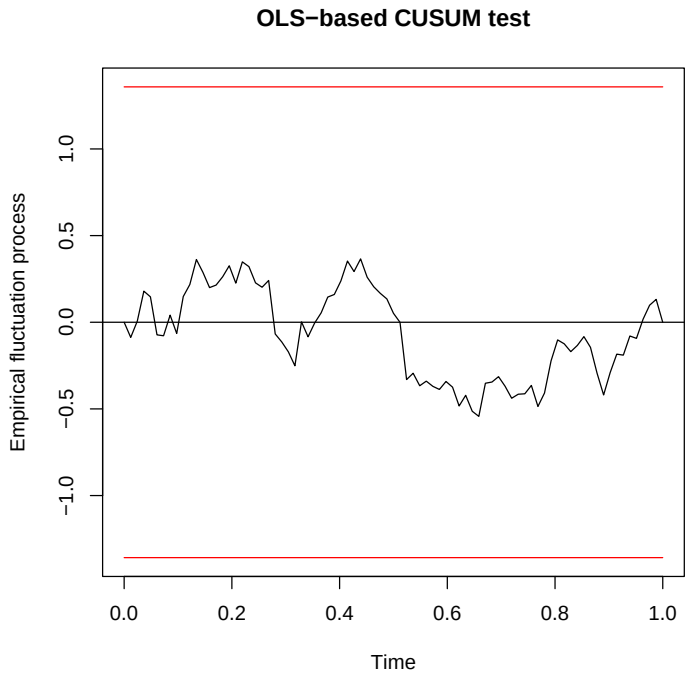

Figure 7: Productivity equation: OLS-Cusum test

$K_1 + K_2$. For the rewritten VAR(p):

$$\begin{bmatrix} \mathbf{y}_{1t} \\ \mathbf{y}_{2t} \end{bmatrix} = \sum_{i=1}^{p} \begin{bmatrix} \alpha_{11,i} & \alpha_{12,i} \\ \alpha_{21,i} & \alpha_{22,i} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{1,t-i} \\ \mathbf{y}_{2,t-i} \end{bmatrix} + CD_t + \begin{bmatrix} \mathbf{u}_{1t} \\ \mathbf{u}_{2t} \end{bmatrix} , \qquad (15)$$

the null hypothesis that the subvector $\mathbf{y}_{1t}$ does not Granger-cause $\mathbf{y}_{2t}$, is defined as $\alpha_{21,i} = 0$ for $i = 1, 2, \ldots, p$. The alternative is: $\exists \alpha_{21,i} \neq 0$ for $i = 1, 2, \ldots, p$. The test statistic is distributed as $F(pK_1K_2, KT - n^*)$, with n^* equal to the total number of parameters in the above VAR(p)-process, including deterministic regressors. The null hypothesis for non-instantaneous causality is defined as: $H_0 : C\sigma = 0$, where C is a $(N \times K(K+1)/2)$ matrix of rank N selecting the relevant co-variances of $\mathbf{u}_{1t}$ and $\mathbf{u}_{2t}$; $\tilde{\sigma} = vech(\tilde{\Sigma}_u)$. The Wald statistic is defined as:

$$\lambda_W = T\tilde{\sigma}'C'[2CD_K^+(\tilde{\Sigma}_u \otimes \tilde{\Sigma}_u)D_K^{+'}C']^{-1}C\tilde{\sigma} , \qquad (16)$$

hereby assigning the Moore-Penrose inverse of the duplication matrix D_K with D_K^+ and $\tilde{\Sigma}_u = \frac{1}{T}\Sigma_{t=1}^{T}\hat{\mathbf{u}}_t\hat{\mathbf{u}}_t'$. The duplication matrix D_K has dimension $(K^2 \times \frac{1}{2}K(K+1))$ and is defined such that for any symmetric $(K \times K)$ matrix A, $vec(A) = D_K vech(A)$ holds. The test statistic λ_W is asymptotically distributed as $\chi^2(N)$.

The function `causality()` is now applied for investigating if the real wage and productivity is causal to employment and unemployment.

```
> causality(var.2c, cause = c("rw", "prod"))
```

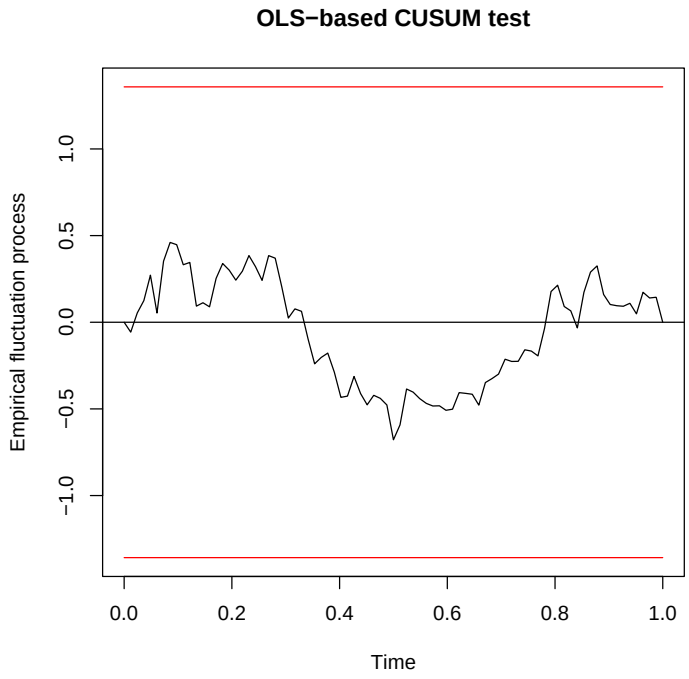

Figure 8: Real wage equation: OLS-Cusum test

```
$Granger

        Granger causality H0: prod rw do not Granger-cause e U

data:  VAR object var.2c
F-Test = 3.4529, df1 = 8, df2 = 292, p-value = 0.0008086


$Instant

        H0: No instantaneous causality between: prod rw and e U

data:  VAR object var.2c
Chi-squared = 2.5822, df = 4, p-value = 0.63
```

The null hypothesis of no Granger-causality from the real wage and labour productivity to employment and unemployment must be rejected; whereas the null hypothesis of non-instantenous causality cannot be rejected. This test outcome is economically plausible, given the frictions observed in labour markets.

2.6 Forecasting

A `predict`-method for objects with class attribute `varest` is available. The `n.ahead` forecasts are computed recursively for the estimated VAR,

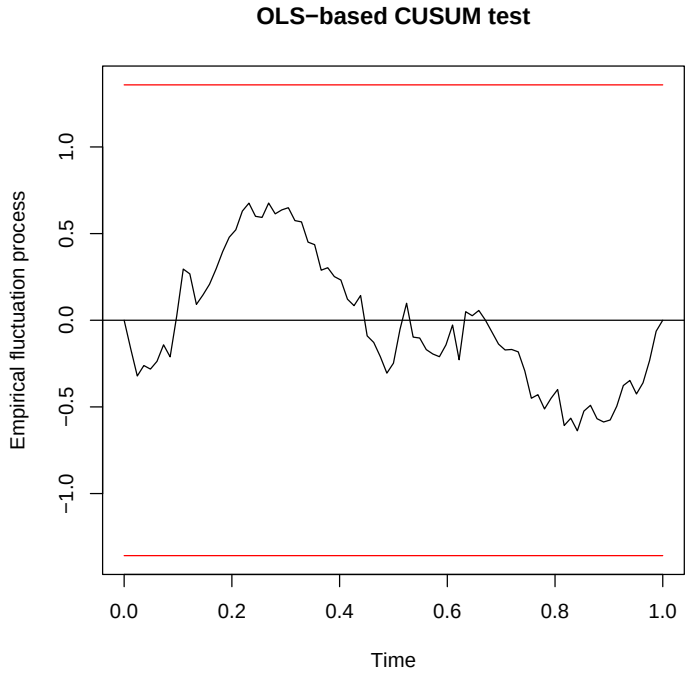

Figure 9: Unemployment equation: OLS-Cusum test

beginning with $h = 1, 2, \ldots, n.ahead$:

$$\mathbf{y}_{T+1|T} = A_1\mathbf{y}_T + \ldots + A_p\mathbf{y}_{T+1-p} + CD_{T+1} \tag{17}$$

```
> var.f10 <- predict(var.2c, n.ahead = 10, ci = 0.95)
> names(var.f10)
[1] "fcst"     "endog"    "model"    "exo.fcst"
> class(var.f10)
[1] "varprd"
> plot(var.f10)
> fanchart(var.f10)
```

Beside the function's arguments for the `varest` object and the `n.ahead` forecast steps, a value for the forecast confidence interval can be provided too. Its default value is 0.95. The forecast error covariance matrix is given as:

$$Cov\left(\begin{bmatrix}\mathbf{y}_{T+1} - \mathbf{y}_{T+1|T}\\ \vdots \\ \mathbf{y}_{T+h} - \mathbf{y}_{T+h|T}\end{bmatrix}\right) = \begin{bmatrix} I & 0 & \cdots & 0 \\ \Phi_1 & I & & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \ldots & I \end{bmatrix} (\Sigma_{\mathbf{u}} \otimes I_h) \begin{bmatrix} I & 0 & \cdots & 0 \\ \Phi_1 & I & & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \ldots & I \end{bmatrix}'$$

and the matrices Φ_i are the coefficient matrices of the Wold moving average representation of a stable VAR(p)-process:

$$\mathbf{y}_t = \Phi_0\mathbf{u}_t + \Phi_1\mathbf{u}_{t-1} + \Phi_2\mathbf{u}_{t-2} + \ldots\ , \tag{18}$$

with $\Phi_0 = I_K$ and Φ_s can be computed recursively according to:

$$\Phi_s = \sum_{j=1}^{s} \Phi_{s-j} A_j \text{ for } s = 1, 2, \ldots, \ , \tag{19}$$

whereby $A_j = 0$ for $j > p$.

The `predict`-method does return a list object of class `varprd` with three elements. The first element, `fcst`, is a list of matrices containing the predicted values, the lower and upper bounds according to the choosen confidence interval, `ci` and its size. The second element, `endog` is a matrix object containing the endogenous variables and the third is the submitted `varest` object. A `plot`-method for objects of class `varprd` does exist as well as a `fanchart()` function for plotting fan charts as described in Britton, Fisher & Whitley [1998].

```
> args(fanchart)

function (x, colors = NULL, cis = NULL)
NULL
```

The `fanchart()` function has `colors` and `cis` arguments, allowing the user to input vectors of colors and critical values. If these arguments are left `NULL`, then as defaults a `heat.map` color scheme is used and the critical values are set from 0.1 to 0.9 with a step size of 0.1. In order to save space, the predict plot for employment and a fan chart for unemployment is shown in the exhibits 10 and 11, respectively.

2.7 Impulse response analysis

The impulse response analysis is based upon the Wold moving average representation of a VAR(p)-process (see equations (18) and (19) above). It is used to investigate the dynamic interactions between the endogenous variables. The $(i, j)th$ coefficients of the matrices Φ_s are thereby interpreted as the expected response of variable $y_{i,t+s}$ to a unit change in variable y_{jt}. These effects can be cumulated through time $s = 1, 2, \ldots$ and hence one would obtain the cumulated impact of a unit change in variable j to the variable i at time s. Aside of these impulse response coefficients, it is often conceivable to use orthogonalised impulse reponses as an alternative. This is the case, if the underlying shocks are less likely to occur in isolation, but rather contemporaneous correlation between the components of the error process $\mathbf{u}_t$ are existent, *i.e.*, the off-diagonal elements of $\Sigma_{\mathbf{u}}$ are non-zero. The orthogonalised impulse responses are derived from a Choleski decomposition of the error variance-covariance matrix: $\Sigma_{\mathbf{u}} = PP'$ with P being a lower triangular. The moving average representation can then be transformed to:

$$\mathbf{y}_t = \Psi_0 \varepsilon_t + \Psi_1 \varepsilon_{t-1} + \ldots \ , \tag{20}$$

with $\varepsilon_t = P^{-1}\mathbf{u}_t$ and $\Psi_i = \Phi_i P$ for $i = 0, 1, 2, \ldots$ and $\Psi_0 = P$. Incidentally, because the matrix P is lower triangular, it follows that only a shock in the first variable of a VAR(p)-process does exert an influence on all the remaining ones and that the second and following variables cannot have a direct impact on y_{1t}. One should bear this in mind whence orthogonal impulse responses are employed (see the code snippet below and figure 12 for a re-ordering of the data to compute the impulse responses originating from a real wage shock to employment and unemployment). Please note

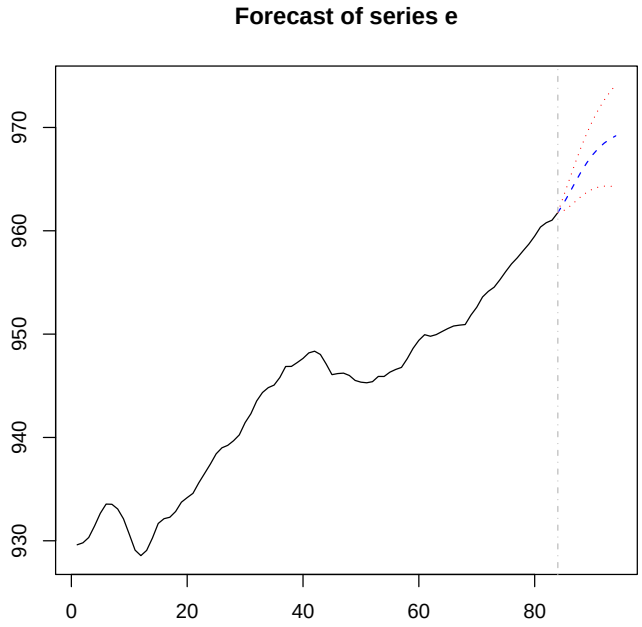

Figure 10: Employment: 10-step ahead forecasts with 95% confidence

further, that a different ordering of the variables might produce different outcomes with respect to the impulse responses. As we shall see in section 3, the non-uniqueness of the impulse reponses can be circumvented by analysing a set of endogenous variables in the structural VAR (SVAR) framework.

```
> args(irf)

function (x, impulse = NULL, response = NULL, n.ahead = 10, ortho = TRUE,
    cumulative = FALSE, boot = TRUE, ci = 0.95, runs = 100, seed = NULL,
    ...)
NULL

> Canada2 <- Canada[, c(3, 1, 4, 2)]
> names(Canada2)

[1] "rw"   "e"    "U"    "prod"

> var.2c.alt <- VAR(Canada2, p = 2, type = "const")
> irf.rw.eU <- irf(var.2c.alt, impulse = "rw", response = c("e",
+     "U"), boot = TRUE)
> names(irf.rw.eU)

 [1] "irf"        "Lower"      "Upper"      "response"   "impulse"
 [6] "ortho"      "cumulative" "runs"       "ci"         "boot"
[11] "model"

> plot(irf.rw.eU)
```

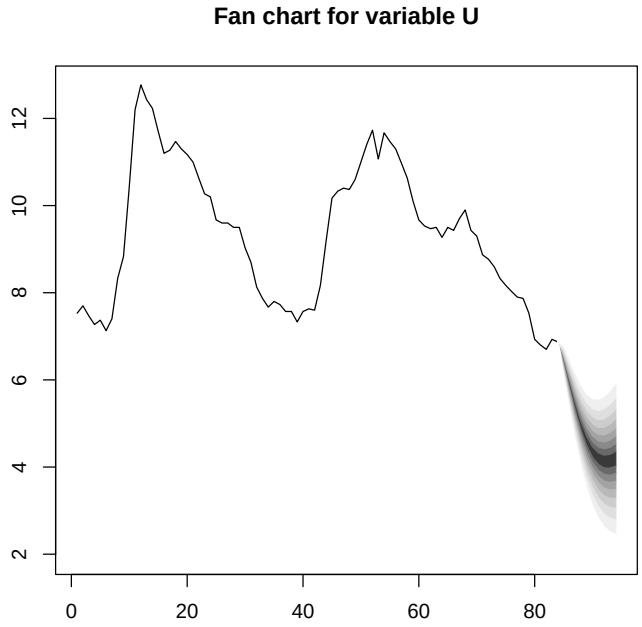

Figure 11: Unemployment: Fan chart with default settings

The function for conducting impulse response analysis is `irf()`. It is a method for objects with class attribute `varest` and `svarest`. The function's arguments are displayed in the R-code example above. The impulse variables are set as a character vector `impulse` and the responses are provided likewise in the argument `response`. If either one is unset, then all variables are considered as impulses or responses, respectively. The default length of the impulse reponses is set to 10 via argument `n.ahead`. The computation of orthogonalised and/or cumulated impulse responses is controlled by the logical switches `ortho` and `cumulative`, respectively. Finally, confidence bands can be returned by setting `boot = TRUE` (default). The pre-set values are to run 100 replications and return 95% confidence bands. It is at the user's leisure to specify a `seed` for replicability of the results. The standard percentile interval is calculated as $CI_s = [s^*_{\gamma/2}, s^*_{(1-\gamma)/2}]$, where $s^*_{\gamma/2}$ and $s^*_{(1-\gamma)/2}$ are the $\gamma/2$ and $(1-\gamma)/2$ quantiles of the estimated bootstraped impulse response coefficients $\hat{\Phi}^*$ or $\hat{\Psi}^*$ (Efron & Tibshirani 1993). The function `irf()` returns an object with class attribute `varirf` for which a `plot` method does exist. In figure 12 the result of the impulse response analysis is depicted. Although a positive real wage shock does influence employment and unemployment in the directions one would *a priori* assume, it does so only significantly for unemployment during the time span ranging from the third period to the sixth period.[2]

[2]Incidentally, the result should not be mistaken or interpreted as demanding higher real wages, *i.e.*, having a free lunch, without facing the consequences beyond of the model's scope.

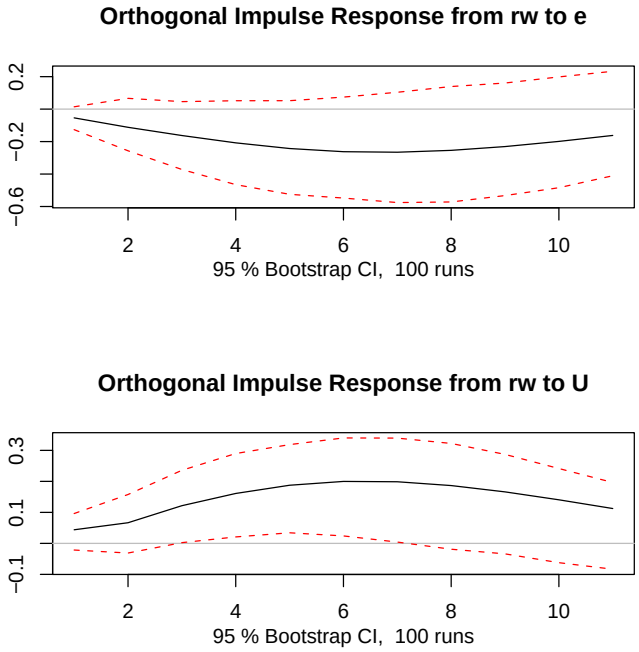

Figure 12: Orthogonal impulse responses from real wage to employment and unemployment (95% confidence bands, boots trapped with 100 replications)

2.8 Forecast error variance decomposition

The forecast error variance decomposition (henceforth: FEVD) is based upon the orthogonalised impulse response coefficient matrices Ψ_n (see section 2.7 above). The FEVD allows the user to analyse the contribution of variable j to the h-step forecast error variance of variable k. If the elementwise squared orthogonalised impulse reponses are divided by the variance of the forecast error variance, $\sigma_k^2(h)$, the resultant is a percentage figure. Formally, the forecast error variance for $y_{k,T+h} - Y_{k,T+h|T}$ is defined as:

$$\sigma_k^2(h) = \sum_{n=0}^{h-1}(\psi_{k1,n}^2 + \ldots + \psi_{kK,n}^2) \tag{21}$$

which can be written as:

$$\sigma_k^2(h) = \sum_{j=1}^{K}(\psi_{kj,0}^2 + \ldots + \psi_{kj,h-1}^2)\ . \tag{22}$$

Dividing the term $(\psi_{kj,0}^2 + \ldots + \psi_{kj,h-1}^2)$ by $\sigma_k^2(h)$ yields the forecast error variance decompositions in percentage terms:

$$\omega_{kj}(h) = (\psi_{kj,0}^2 + \ldots + \psi_{kj,h-1}^2)/\sigma_k^2(h)\ . \tag{23}$$

The method `fevd` is available for conducting FEVD. Functions for objects of classes `varest` and `svarest` do exist. The argument aside of an object

The example has been primarily choosen for demonstrations purposes only.

with class attribute being either `varest` or `svarest` is the number of forecasting steps, `n.ahead`. Its default value has been set to 10. Currently, the ...-argument is unused.

```
> args(fevd)

function (x, n.ahead = 10, ...)
NULL
```

The method does return a list object with class attribute `varfevd` for which a `plot`-method does exists. The list elements are the forecast error variances organised on a per-variable basis.

```
> var2c.fevd <- fevd(var.2c, n.ahead = 5)
> class(var2c.fevd)

[1] "varfevd"

> names(var2c.fevd)

[1] "e"    "prod" "rw"   "U"

> var2c.fevd$e

             e       prod          rw           U
[1,] 1.0000000 0.00000000 0.000000000 0.000000000
[2,] 0.9633815 0.02563062 0.004448081 0.006539797
[3,] 0.8961692 0.06797131 0.013226872 0.022632567
[4,] 0.8057174 0.11757589 0.025689192 0.051017495
[5,] 0.7019003 0.16952744 0.040094324 0.088477959
```

From the example above, it is evident that aside of employment itself, the influence of productivity is contributing the most to the forecast uncertainty of employment. In the exhibit 13, the FEVD of employment for five ahead steps is plotted as a bar graph.[3]

3 SVAR: Structural vector autoregressive models

3.1 Definition

Recall from section 2.1 on page 2 the definition of a VAR(p)-process, in particular equation (1). A VAR(p) can be interpreted as a reduced form model. A SVAR-model is its structural form and is defined as:

$$A\mathbf{y}_t = A_1^*\mathbf{y}_{t-1} + \ldots + A_p^*\mathbf{y}_{t-p} + B\varepsilon_t \ . \tag{24}$$

It is assumed that the *structural errors*, ε_t, are white noise and the coefficient matrices A_i^* for $i = 1, \ldots, p$, are structural coefficients that might differ from their reduced form counterparts. To see this, consider the resulting equation by left-multiplying equation (24) with the inverse of A:

$$\begin{aligned} \mathbf{y}_t &= A^{-1}A_1^*\mathbf{y}_{t-1} + \ldots + A^{-1}A_p^*\mathbf{y}_{t-p} + A^{-1}B\varepsilon_t \\ \mathbf{y}_t &= A_1\mathbf{y}_{t-1} + \ldots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t \ . \end{aligned} \tag{25}$$

[3] If the `plot`-method is called interactively, the user is enforced to go through the bar graphs for all variables. In order to save space, only the FEVD for the employment variable is shown in figure 13.

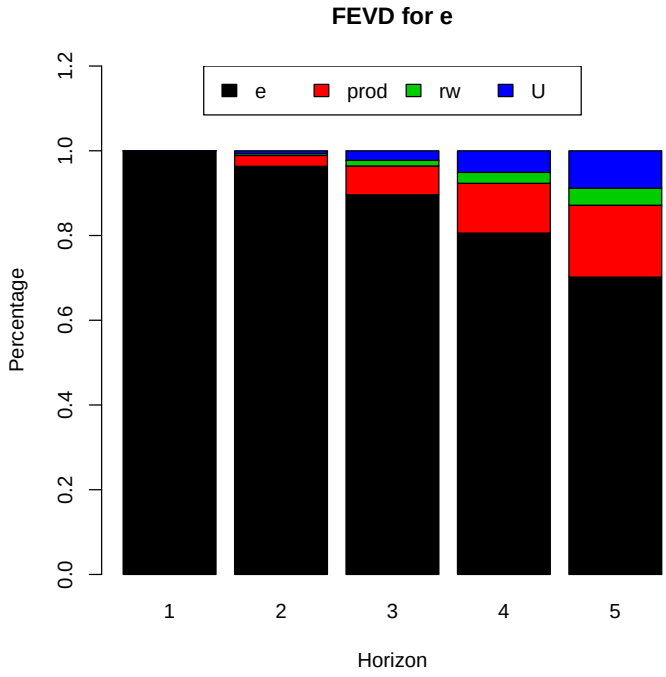

Figure 13: Forecast error variance decomposition for employment

A SVAR-model can be used to identify shocks and trace these out by employing IRA and/or FEVD through imposing restrictions on the matrices A and/or B. Incidentally, though a SVAR-model is a structural model, it departs from a reduced form VAR(p)-model and only restrictions for A and B can be added. It should be noted that the reduced form residuals can be retrieved from a SVAR-model by $\mathbf{u}_t = A^{-1}B\varepsilon_t$ and its variance-covariance matrix by $\Sigma_{\mathbf{u}} = A^{-1}BB'A^{-1'}$.

Depending on the imposed restrictions, three types of SVAR- models can be distinguished:

- A model: B ist set to I_K (minimum number of restrictions for identification is $K(K-1)/2$).
- B model: A ist set to I_K (minimum number of restrictions to be imposed for identification is the same as for A model).
- AB model: restrictions can be placed on both matrices (minimum number of restrictions for identification is $K^2 + K(K-1)/2$).

3.2 Estimation

A SVAR model is estimated with the function `SVAR()`.[4] Its arguments are shown in the R-code example below. An object with class attribute

[4]The exhibition is confined to the function `SVAR()`. Incidentally, with function `SVAR2()` a SVAR model can estimated too, thereby a scoring algorithm as suggested by Amisano & Giannini [1997] is utilised.

`varest` has to be provided as argument `x`. Whether an A-, B- or AB-model will be estimated, is dependent on the setting for `Amat` and `Bmat`. If a restriction matrix for `Amat` with dimension $(K \times K)$ is provided and the argument `Bmat` is left `NULL`, an A-model be estimated. In this case `Bmat` is set to an identity matrix I_K. Alternatively, if only a matrix object for `Bmat` is provided and `Amat` left unchanged, then a B-model will be estimated and internally `Amat` is set to an identity matrix I_K. Finally, if matrix objects for both arguments are provided, then an AB-model will be estimated. In all cases, the matrix elements to be estimated are marked by `NA` entries at the relevant positions.

```
> args(SVAR)

function (x, Amat = NULL, Bmat = NULL, start = NULL, ...)
NULL

> amat <- diag(4)
> diag(amat) <- NA
> amat[1, 2] <- NA
> amat[1, 3] <- NA
> amat[3, 2] <- NA
> amat[4, 1] <- NA
> amat

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

In the example above, an expository object `amat` has been set up. The next function's argument is `start`. This argument expects a vector of starting values, hence, the user has one handle at hand for controlling the optimisation process in case one encounters convergence problems. If left `NULL`, then starting values are set to 0.1 for all coefficients to be optimised. Finally, the ... argument in `SVAR()` is parsed to the `optim()` function. Therefore, it is possible to choose the numerical optimisation method and set `hessian = TRUE` to obtain the numerical standard errors, for instance.

The parameters are estimated by minimising the negative of the concentrated log-likelihood function:

$$\begin{aligned} \ln L_c(A, B) = &- \frac{KT}{2}\ln(2\pi) + \frac{T}{2}\ln|A|^2 - \frac{T}{2}\ln|B|^2 \\ &- \frac{T}{2} tr(A'B'^{-1}B^{-1}A\tilde{\Sigma}_u) \,, \end{aligned} \tag{26}$$

by utilising `optim`.

```
> args(optim)

function (par, fn, gr = NULL, method = c("Nelder-Mead", "BFGS",
    "CG", "L-BFGS-B", "SANN"), lower = -Inf, upper = Inf, control = list(),
    hessian = FALSE, ...)
NULL

> svar2c.A <- SVAR(var.2c, Amat = amat, Bmat = NULL, hessian = TRUE,
+     method = "BFGS")
> svar2c.A
```

```
#################
# SVAR: A-model #
#################


LR overidentification test:

        LR overidentification

data:  var.2c
Chi^2 = 0.8252, df = 2, p-value = 0.6619


Estimated A matrix:
            e        prod        rw        U
e    2.787720  0.01998724 0.1906375 0.000000
prod 0.000000  1.53260287 0.0000000 0.000000
rw   0.000000 -0.19609242 1.2920240 0.000000
U    2.562603  0.00000000 0.0000000 4.882572

Estimated standard errors for A matrix:
             e      prod        rw        U
e    0.2176737 0.1706396 0.1434537 0.000000
prod 0.0000000 0.1196780 0.0000000 0.000000
rw   0.0000000 0.1699440 0.1008898 0.000000
U    0.3642575 0.0000000 0.0000000 0.381258

Estimated B matrix:
     e prod rw U
e    1    0  0 0
prod 0    1  0 0
rw   0    0  1 0
U    0    0  0 1

Estimated standard errors for B matrix:
     e prod rw U
e    0    0  0 0
prod 0    0  0 0
rw   0    0  0 0
U    0    0  0 0
```

The returned object of function `SVAR()` is a list with class attribute `svarest`.

```
> class(svar2c.A)

[1] "svarest"

> names(svar2c.A)

 [1] "A"       "Ase"     "B"       "Bse"     "LRIM"    "Sigma.U" "LR"
 [8] "opt"     "start"   "type"    "var"     "call"
```

Dependent on the choosen model and if the argument `hessian = TRUE` has been set, the list elements `A, Ase, B, Bse` contain the estimated coefficient matrices with the numerical standard errors, if applicable. The element `LRIM` does contain the long-run impact matrix in case a SVAR of type Blanchard & Quah is estimated, otherwise this element is `NULL`

(see below for more details). The list element `Sigma.U` is the variance-covariance matrix of the reduced form residuals times 100, *i.e.*, $\Sigma_U = A^{-1}BB'A^{-1'} \times 100$. The list element `LR` is an object with class attribute `htest`, holding the Likelihood ratio overidentification test, whereby the test statistic is defined as:

$$LR = T(\log det(\tilde{\Sigma}_{\mathbf{u}}^{r}) - \log det(\tilde{\Sigma}_{\mathbf{u}})) \tag{27}$$

with $\tilde{\Sigma}_{\mathbf{u}}$ being the ML estimator of the reduced form variance-covariance matrix and $\tilde{\Sigma}_{\mathbf{u}}^{r}$ is the corresponding estimator obtained from the restricted structural form estimation. The element `opt` is the returned object from function `optim`. The remaining four list items are the vector of starting values, the SVAR model type, the `varest` object and the `call` to `SVAR()`.

Finally, it should be noted that the model proposed by Blanchard & Quah [1989] can be considered as a particular SVAR-model of type B. The matrix A is equal to I_K and the matrix of the long-run effects is lower-triangular and defined as:

$$(I_K - A_1 - \cdots - A_P)^{-1}B \quad . \tag{28}$$

Hence, the residual of the second equation cannot exert a long-run influence on the first variable and likewise the third residual cannot impact the first and second variable. The estimation of the Blanchard & Quah model is achieved by a Choleski decomposition of:

$$(I_K - \hat{A}_1 - \cdots - \hat{A}_p)^{-1}\hat{\Sigma}_u(I_K - \hat{A}_1' - \cdots - \hat{A}_p')^{-1} \quad . \tag{29}$$

The matrices $\hat{A}_i$ for $i = 1, \ldots, p$ assign the reduced form estimates. The long-run impact matrix is the lower-triangular Choleski decomposition of this matrix and the contemporaneous impact matrix is equal to:

$$(I_K - A_1 - \cdots - A_p)Q \quad , \tag{30}$$

where Q assigns the lower-trinagular Choleski decomposition. It should be noted that a re-ordering of the VAR might become necessary due to the recursive nature of the model.

The Blanchard & Quah model is implemented as function `BQ()`. It has one argument x, which is an object with class attribute `varest`. The function does return a list object with class attribute `svarest`. Below is a trivial example of applying a Blanchard & Quah [1989]-type SVAR to the already available VAR `var.2c`.

```
> BQ(var.2c)

########################
# SVAR: Blanchard-Quah #
########################


Estimated contemporaneous impact matrix:
              e        prod          rw           U
e    -0.00764432 -0.28469582  0.07374319 -0.21233590
prod  0.54366334  0.21657828 -0.03379321 -0.28651841
rw    0.08211181  0.28588183  0.71874239  0.06161939
U     0.12945102  0.05667792 -0.01039129  0.24110588
```

```
Estimated identified long run impact matrix:
             e       prod        rw         U
e    104.37389  0.0000000  0.000000 0.0000000
prod  45.35215  5.1971134  0.000000 0.0000000
rw   168.40969 -2.1144696 10.719506 0.0000000
U    -19.25842 -0.4561694  1.410200 0.5331401
```

The contemporaneous impact matrix is stored as list-element `B` and the long-run impact matrix as list-element `LRIM` in the returned object.

3.3 Impulse response analysis

Like impulse response analysis can be conducted for objects with class attribute `varest` it can be done so for objects with class attribute `svarest` (see section 2.7 on page 20 following). In fact, the `irf`-methods for classes `varest` and `svarest` are at hand with the same set of arguments; except `ortho` is missing for objects of class `svarest` due to the nature and interpretation of the error terms in a SVAR. The impulse response coefficients for a SVAR are calculated as $\Theta_i = \Phi_i A^{-1} B$ for $i = 1, \ldots, n$.
In the code snippet below, an example of an IRA is exhibited for the estimated SVAR from the previous section.

```
> svar2cA.ira <- irf(svar2c.A, impulse = "rw", response = c("e",
+     "U"), boot = FALSE)
> svar2cA.ira

Impulse response coefficients
$rw
                e          U
 [1,] -0.05292837 0.02777929
 [2,] -0.12816194 0.06237647
 [3,] -0.19908356 0.13215216
 [4,] -0.26547797 0.18578989
 [5,] -0.32043081 0.22492294
 [6,] -0.35743849 0.24753946
 [7,] -0.37478736 0.25405194
 [8,] -0.37389275 0.24686257
 [9,] -0.35784064 0.22915640
[10,] -0.33036781 0.20422207
[11,] -0.29519673 0.17504960
```

3.4 Forecast error variance decomposition

A forecast error variance decomposition can be applied likewise to objects of class `svarest`. Here, the forecast errors of $y_{T+h|T}$ are derived from the impulse responses of a SVAR and the derivation to the forecast error variance decomposition is similar to the one outlined for VARs (see section 2.8 on page 22 following).
An application is provided in the code snippet below and a plot for the real wage equation is exhibited in figure 14 on page 30.

```
> svar2cA.fevd <- fevd(svar2c.A, n.ahead = 8)
> plot(svar2cA.fevd)
```

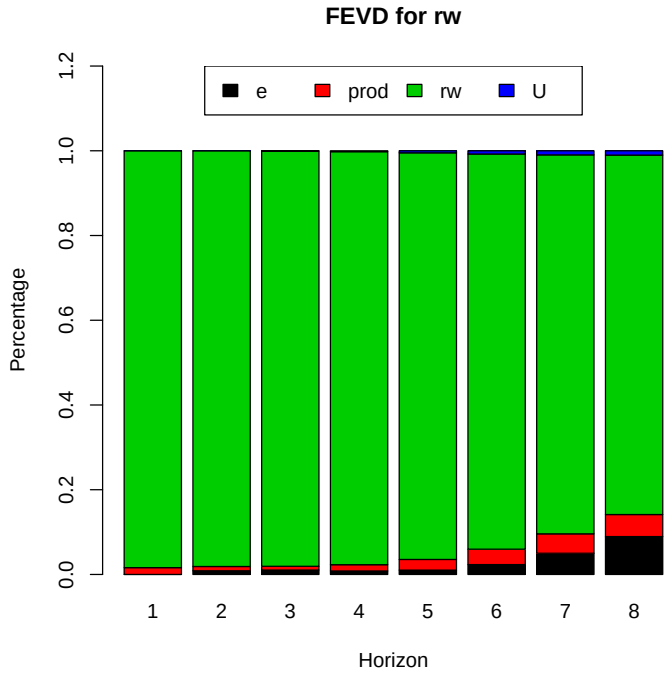

Figure 14: SVAR: FEVD for real wage

4 VECM to VAR

Reconsider the VAR presentation from equation 1 on page 2:

$$\mathbf{y}_t = A_1\mathbf{y}_{t-1} + \ldots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t \quad , \tag{31}$$

The following vector error correction specifications do exist, which can be estimated with function `ca.jo()` contained in `urca`:

$$\Delta\mathbf{y}_t = \Pi\mathbf{y}_{t-p} + \Gamma_1\Delta\mathbf{y}_{t-1} + \ldots + \Gamma_{p-1}\mathbf{y}_{t-p+1} + \mathbf{u}_t \quad , \tag{32}$$

with

$$\Gamma_i = -(I - A_1 - \ldots - A_i), \quad i = 1, \ldots, p-1. \tag{33}$$

and

$$\Pi = -(I - A_1 - \ldots - A_p) \quad . \tag{34}$$

The Γ_i matrices contain the cumulative long-run impacts, hence this VECM specification is signified by longrun form. The other specification is given as follows:

$$\Delta\mathbf{y}_t = \Pi\mathbf{y}_{t-1} + \Gamma_1\Delta\mathbf{y}_{t-1} + \ldots + \Gamma_{p-1}\mathbf{y}_{t-p+1} + \mathbf{u}_t \quad , \tag{35}$$

with

$$\Gamma_i = -(A_{i+1} + \ldots + A_p) \quad i = 1, \ldots, p-1. \tag{36}$$

Equation 34 applies to this specification too. Hence, the Π matrix is the same as in the first specification. However, the Γ_i matrices now differ, in the sense that they measure transitory effects. Therefore this specification

is signified by transitory form.

With function `vec2var()` a VECM (*i.e* an object of formal class `ca.jo`, generated by `ca.jo()`) is transormed into its level VAR representation. For illustration purposes, an example taken from Johansen & Juselius [1990] is used in the following.

```
> library(urca)
> data(finland)
> sjf <- finland
> sjf.vecm <- ca.jo(sjf, constant = FALSE, type = "eigen", K = 2,
+     spec = "longrun", season = 4, ctable = "A2")
> summary(sjf.vecm)

######################
# Johansen-Procedure #
######################

Test type: maximal eigenvalue statistic (lambda max) , with linear trend

Eigenvalues (lambda):
[1] 0.30932660 0.22599561 0.07308056 0.02946699

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 3 |  3.11  6.69  8.08 11.58
r <= 2 |  7.89 12.78 14.60 18.78
r <= 1 | 26.64 18.96 21.28 26.15
r = 0  | 38.49 24.92 27.34 32.62

Eigenvectors, normalised to first column:
(These are the cointegration relations)

           lrm1.l2    lny.l2    lnmr.l2    difp.l2
lrm1.l2  1.0000000  1.000000  1.0000000   1.000000
lny.l2  -0.9763252 -1.323191 -0.9199865   1.608739
lnmr.l2 -7.0910749 -2.016033  0.2691516  -1.375342
difp.l2 -7.0191097 22.740851 -1.8223931 -15.686927

Weights W:
(This is the loading matrix)

           lrm1.l2       lny.l2      lnmr.l2      difp.l2
lrm1.d 0.033342108 -0.020280528 -0.129947614 -0.002561906
lny.d  0.022544782 -0.005717446  0.012949130 -0.006265406
lnmr.d 0.053505000  0.046876449 -0.007367715  0.002173242
difp.d 0.005554849 -0.017353903  0.014561151  0.001531004
```

One can conclude that two cointegration relationships do exist. The cointegrating rank is needed for the above transformation with function `vec2var`. Given the object `sjf.vecm` of class `ca.jo` with $r = 2$, one can now swiftly proceed:

```
> args(vec2var)
```

```
function (z, r = 1)
NULL
```

```
> sjf.var <- vec2var(sjf.vecm, r = 2)
> sjf.var
```

```
Coefficient matrix of lagged endogenous variables:

A1:
          lrm1.l1      lny.l1     lnmr.l1    difp.l1
lrm1  0.855185363 -0.28226832 -0.09298924 -0.1750511
lny   0.036993826  0.33057494 -0.06731145 -0.1946863
lnmr -0.156875074 -0.01067717  0.76861874  0.4247362
difp  0.001331951  0.02850137  0.02361709  0.2063468


A2:
         lrm1.l2      lny.l2     lnmr.l2     difp.l2
lrm1  0.15787622  0.27655060 -0.10255593 -0.52017728
lny  -0.02016649  0.65497929 -0.08102873 -0.09357761
lnmr  0.25725652 -0.10358761 -0.24253117  0.26571672
difp -0.01313100 -0.01096218 -0.02802090  0.36002057


Coefficient matrix of deterministic regressor(s).

        constant         sd1          sd2         sd3
lrm1  0.03454360 0.039660747  0.037177941  0.10095683
lny   0.05021877 0.043686282  0.082751819  0.09559270
lnmr  0.22729778 0.008791390  0.012456612  0.02011396
difp -0.03055891 0.001723883 -0.007525805 -0.00835411
```

The `print` method does return the coefficient values, first for the lagged endogenous variables, next for the deterministic regressors. The returned object is of class `vec2var` and this list has the following elements:

```
> names(sjf.var)
```

```
 [1] "deterministic" "A"             "p"             "K"
 [5] "y"             "obs"           "totobs"        "call"
 [9] "vecm"          "datamat"       "resid"         "r"
```

```
> class(sjf.var)
```

```
[1] "vec2var"
```

```
> methods(class = "vec2var")
```

```
 [1] fevd.vec2var*      fitted.vec2var*    irf.vec2var*       logLik.vec2var*
 [5] Phi.vec2var*       plot.vec2var*      predict.vec2var*   print.vec2var*
 [9] Psi.vec2var*       residuals.vec2var*

   Non-visible functions are asterisked
```

The object has assigned class attribute `vec2var` and a `print-` , `irf-`, `fevd-` , `predict-`, `Phi` and `Psi`-method do exist. Furthermore, the functions

`arch()`, `normality()`, `serial()` and `fanchart()` can be employed for further analysis too.[5]

Finally, structural vector error correction models can be estimated with the function `SVEC()`. Likewise to the function `SVAR2()` a scoring algorithm is utilised for determining the coefficients of the long run and contemporaneous impact matrices. The estimated standard errors of these parameters can be calculated by a bootstrapping procedure. The latter is controlled by the functional argument `boot` in `SVEC`; its default value is `FALSE`.

List of Tables

1 Regression result for employment equation 5
2 Regression result for productivity equation 7
3 Regression result for real wage equation 7
4 Regression result for unemployment equation 10

List of Figures

1 Canada: Macroeconomic series 4
2 Employment equation: Diagram of fit, residuals with ACF and PACF 6
3 Productivity equation: Diagram of fit, residuals with ACF and PACF 8
4 Real wage equation: Diagram of fit, residuals with ACF and PACF 9
5 Unemployment equation: Diagram of fit, residuals with ACF and PACF 11
6 Employment equation: OLS-Cusum test 16
7 Productivity equation: OLS-Cusum test 17
8 Real wage equation: OLS-Cusum test 18
9 Unemployment equation: OLS-Cusum test 19
10 Employment: 10-step ahead forecasts with 95% confidence . 20
11 Unemployment: Fan chart with default settings 21
12 Orthogonal impulse responses from real wage to employment and unemployment (95% confidence bands, boots trapped with 100 replications) 23
13 Forecast error variance decomposition for employment . . . 25
14 SVAR: FEVD for real wage 30

References

Amisano, G. & Giannini, C. [1997], *Topics in Structural VAR Econometrics*, 2 edn, Springer, Berlin.

Banerjee, A., Dolado, J., Galbraith, J. & Hendry, D. [1993], *Co-Integration, Error-Correction, and the Econometric Analysis of Non-Stationary Data*, Oxford University Press.

[5] The reader is referred to the previous sections in which these methods and functions are outlined.

Bera, A. K. & Jarque, C. M. [1980], 'Efficient tests for normality, homoscedasticity and serial independence of regression residuals', *Economics Letters* **6**(3), 255–259.

Bera, A. K. & Jarque, C. M. [1981], 'Efficient tests for normality, homoscedasticity and serial independence of regression residuals: Monte carlo evidence', *Economics Letters* **7**(4), 313–318.

Blanchard, O. & Quah, D. [1989], 'The dynamic effects of aggregate demand and supply disturbances', *The American Economic Review* **79**(4), 655–673.

Brandt, P. [2006], *MSBVAR: Bayesian Vector Autoregression Models, Impulse Responses and Forecasting.* R package version 0.1.1.
URL: *http://www.utdallas.edu/ pbrandt/*

Breusch, T. [1978], 'Testing for autocorrelation in dynamic linear models', *Australien Economic Papers* **17**, 334–355.

Britton, E., Fisher, P. & Whitley, J. [1998], 'The inflation report projections: understanding the fan chart', *Bank of England Quarterly Bulletin* **38**, 30–37.

Edgerton, D. & Shukur, G. [1999], 'Testing autocorrelation in a system perspective', *Econometric Reviews* **18**, 343–386.

Efron, B. & Tibshirani, R. [1993], *An Introduction to the Bootstrap*, Chapman & Hall, New York.

Engle, R. [1982], 'Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation', *Econometrica* **50**, 987–1007.

Engle, R. & Granger, C. [1987], 'Co-integration and error correction: Representation, estimation and testing', *Econometrica* **55**, 251–276.

Gilbert, P. [1993], State space and arma models: An overview of the equivalence, Working Paper 93-4, Bank of Canada, Ottawa, Canada.
URL: *http:www.bank-banque-canada.ca/pgilbert/*

Gilbert, P. [1995], 'Combining var estimation and state space model reduction for simple good predictions', *Journal of Forecasting: Special Issue on VAR Modelling* **14**, 229–250.

Gilbert, P. [2000], 'A note on the computation of time series model roots', *Applied Economics Letters* **7**, 423–424.

Godfrey, L. [1978], 'Testing for higher order serial correlation in regression equations when the regressors include lagged dependent variables', *Econometrica* **46**, 1303–1310.

Granger, C. [1969], 'Investigating causal relations by econometric models and cross-spectral methods', *Econometrica* **37**, 424–438.

Granger, C. [1981], 'Some properties of time series data and their use in econometric model specification', *Journal of Econometrics* **16**, 121–130.

Hamilton, J. D. [1994], *Time Series Analysis*, Princeton University Press.

Hendry, D. [1995], *Dynamic Econometrics*, Oxford University Press, Oxford.

Jarque, C. M. & Bera, A. K. [1987], 'A test for normality of observations and regression residuals', *International Statistical Review* **55**, 163–172.

Johansen, S. [1995], *Likelihood Based Inference in Cointegrated Vector Autoregressive Models*, Oxford University Press, Oxford.

Johansen, S. & Juselius, K. [1990], 'Maximum likelihood estimation and inference on cointegration – with applications to the demand for money', *Oxford Bulletin of Economics and Statistics* **52**(2), 169–210.

Lütkepohl, H. [2006], *New Introduction to Multiple Time Series Analysis*, Springer.

Lütkepohl, H. & Breitung, J. [1997], Impulse response analysis of vector autoregressive processes, *in* C. Heij, H. Schumacher, B. Hanzon & C. Praagman, eds, 'System Dynamics in Economic and Financial Models', John Wiley, Chichester.

Lütkepohl, H., Krätzig, M., Phillips, P., Ghysels, E. & Smith, R. [2004], *Applied Time Series Econometrics (Themes in Modern Econometrics)*, Cambridge University Press.

Pfaff, B. [2006], *Analysis of Integrated and Cointegrated Time Series with R*, first edn, Springer, New York. ISBN 0-387-27960-1.
URL: *http://www.pfaffikus.de*

Sims, C. [1980], 'Macroeconomics and reality', *Econometrica* **48**, 1–48.

Venables, W. & Ripley, B. [2002], *Modern Applied Statistics with S*, fourth edn, Springer, New York. ISBN 0-387-95457-0.
URL: *http://www.stats.ox.ac.uk/pub/MASS4/*

Watson, M. [1994], Vector autoregressions and cointegration, *in* 'Handbook of Econometrics', Vol. IV, Elsevier, New York.

Würtz, D. [2006], *fMultivar: Rmetrics - Multivariate Market Analysis*. R package version 221.10065.
URL: *http://www.rmetrics.org*

Zeileis, A., Leisch, F., Hornik, K. & Kleiber, C. [2002], 'strucchange: An r package for testing for structural change in linear regression models', *Journal of Statistical Software* **7**(2), 1–38.
URL: *http://www.jstatsoft.org/v07/i02/*