

Multivariate Analysis of Stationary Time Series

This is the second chapter that presents models confined to stationary time series, but now in the context of multivariate analysis. Vector autoregressive models and structural vector autoregressive models are introduced. The analytical tools of impulse response functions, forecast error variance decomposition, and Granger causality, as well as forecasting and diagnostic tests, are outlined. As will be shown later, these concepts can be applied to cointegrated systems, too.

2.1 Overview

Since the critique of Sims [1980] in the early 1980s, VAR analysis has evolved as a standard instrument in econometrics for analyzing multivariate time series. Because statistical tests are highly used in determining interdependence and dynamic relationships between variables, it soon became evident that this methodology could be enriched by incorporating non-statistical *a priori* information; hence SVAR models evolved that try to bypass these shortcomings. These kinds of models are considered in Section 2.3. At the same time as Sims jeopardized the paradigm of multiple structural equation models laid out by the Cowles Foundation in the 1940s and 1950s, Granger [1981] and Engle and Granger [1987] endowed econometricians with a powerful tool for modeling and testing economic relationships, namely the concept of integration and cointegration. Nowadays these traces of research are unified in the form of *vector error-correction* and *structural vector error-correction models*. These topics are deferred to Chapters 4 and 8.

2.2 Vector Autoregressive Models

2.2.1 Specification, Assumptions, and Estimation

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

$$\mathbf{y}_t = A_1 \mathbf{y}_{t-1} + \dots + A_p \mathbf{y}_{t-p} + CD_t + \mathbf{u}_t, \quad (2.1)$$

where A_i are $(K \times K)$ coefficient matrices for $i = 1, \dots, p$ and \mathbf{u}_t is a K -dimensional white noise process with time-invariant positive definite covari-

ance matrix $E(\mathbf{u}_t \mathbf{u}_t') = \Sigma_{\mathbf{u}}$. The matrix C is the coefficient matrix of potentially deterministic regressors with dimension $(K \times M)$, and D_t is an $(M \times 1)$ column vector holding the appropriate deterministic regressors, such as a constant, trend, and dummy and/or seasonal dummy variables.

Equation (2.1) is sometimes written in the form of a lag polynomial $A(L) = (I_K - A_1 - \dots - A_p)$ as

$$A(L)\mathbf{y}_t = CD_t + \mathbf{u}_t. \quad (2.2)$$

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

$$\det(I_K - A_1 z - \dots - A_p z^p) \neq 0 \quad \text{for } |z| \leq 1. \quad (2.3)$$

If the solution of the preceding 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.*, $I(1)$), a topic of the next chapter.

In practice, the stability of an empirical VAR(p)-process can be analyzed by considering the companion form and calculating the *eigenvalues* of the coefficient matrix (see Lütkepohl [2006] for a detailed derivation). A VAR(p)-process can be written as a VAR(1)-process as

$$\xi_t = A\xi_{t-1} + \mathbf{v}_t \quad (2.4)$$

with

$$\xi_t = \begin{bmatrix} \mathbf{y}_t \\ \vdots \\ \mathbf{y}_{t-p+1} \end{bmatrix}, \quad A = \begin{bmatrix} A_1 & A_2 & \cdots & A_{p-1} & A_p \\ I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}, \quad \mathbf{v}_t = \begin{bmatrix} \mathbf{u}_t \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}, \quad (2.5)$$

where the dimension of the stacked vectors ξ_t and \mathbf{v}_t is $(Kp \times 1)$ and that 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. For a given sample of the endogenous variables $\mathbf{y}_1, \dots, \mathbf{y}_T$ and sufficient presample values $\mathbf{y}_{-p+1}, \dots, \mathbf{y}_0$, the coefficients of a VAR(p)-process can be estimated efficiently by least squares applied separately to each of the equations. If the error process \mathbf{u}_t is normally distributed, then this estimator is equal to the maximum likelihood estimator conditional on the initial values.

It was shown in the previous chapter that a stable AR(p)-process can be represented as an infinite MA-process (see Equations (1.10a) and (1.10b)). This result applies likewise to a stable VAR(p)-process. Its *Wold moving average representation* is given as

$$\mathbf{y}_t = \Phi_0 \mathbf{u}_t + \Phi_1 \mathbf{u}_{t-1} + \Phi_2 \mathbf{u}_{t-2} + \dots \quad (2.6)$$

with $\Phi_0 = I_K$, and the Φ_s matrices can be computed recursively according to

$$\Phi_s = \sum_{j=1}^s \Phi_{s-j} A_j \quad \text{for } s = 1, 2, \dots, \quad (2.7)$$

where $\Phi_0 = I_K$ and $A_j = 0$ for $j > p$.

Before considering an artificial data set, one topic should be touched on first, namely the empirical determination of an appropriate lag order. As in the univariate AR(p)-models, the lag length can be determined by *information criteria* such as those of Akaike [1981], Hannan and Quinn [1979], Quinn [1980], or Schwarz [1978], or by the *final prediction error* (see Lütkepohl [2006] for a detailed exposition of these criteria). These measures are defined as

$$\text{AIC}(p) = \log \det(\tilde{\Sigma}_u(p)) + \frac{2}{T} pK^2, \quad (2.8a)$$

$$\text{HQ}(p) = \log \det(\tilde{\Sigma}_u(p)) + \frac{2 \log(\log(T))}{T} pK^2, \quad (2.8b)$$

$$\text{SC}(p) = \log \det(\tilde{\Sigma}_u(p)) + \frac{\log(T)}{T} pK^2, \text{ or} \quad (2.8c)$$

$$\text{FPE}(p) = \left(\frac{T + p^*}{T - p^*} \right)^K \det(\tilde{\Sigma}_u(p)), \quad (2.8d)$$

with $\tilde{\Sigma}_u(p) = T^{-1} \sum_{t=1}^T \hat{\mathbf{u}}_t \hat{\mathbf{u}}_t'$, and p^* is the total number of parameters in each equation and p assigns the lag order. It is shown in Lütkepohl [2006] that $\ln(\text{FPE})$ and AIC will indicate similar lag orders for moderate and large sample sizes. The following relations can be further deduced:

$$\hat{p}(\text{SC}) \leq \hat{p}(\text{AIC}) \quad \text{if } T \geq 8, \quad (2.9a)$$

$$\hat{p}(\text{SC}) \leq \hat{p}(\text{HQ}) \quad \text{for all } T, \quad (2.9b)$$

$$\hat{p}(\text{HQ}) \leq \hat{p}(\text{AIC}) \quad \text{if } T \geq 16. \quad (2.9c)$$

These information criteria are implemented in the functions `VAR()` and `VARselect()` contained in the package `vars`.¹ In the former function, an appropriate VAR(p)-model will be estimated by providing the maximal lag number, `lag.max`, and the desired criterion. The calculations are based upon the same sample size. That is, `lag.max` values are used as starting values for each of the estimated models. The result of the function `VARselect()` is a list object with elements `selection` and `criteria`. The element `selection` is a vector of optimal lag length according to the above-mentioned information criteria. The element `criteria` is a matrix containing the particular values for each of these criteria up to the maximal lag order chosen.

¹ The package `vars` can be obtained from CRAN, and it is hosted on R-Forge as project AICTS II; see <http://CRAN.r-project.org> and <http://r-forge.r-project.org/projects/vars/>, respectively.

Table 2.1. VAR result for y_1

Variable	Estimate	Std. Error	t -value	$\Pr(> t)$
Lagged levels				
y1.l1	0.4998	0.0354	14.1003	0e + 00
y2.l1	0.1551	0.0407	3.8085	2e - 04
y1.l2	-0.3291	0.0352	-9.3468	0e + 00
y2.l2	-0.7550	0.0454	-16.6466	0e + 00
Deterministic				
const.	5.9196	0.6197	9.5531	0e + 00

We will now generate an artificial two-dimensional VAR(2)-process that obeys the following form:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} 5.0 \\ 10.0 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.2 \\ -0.2 & -0.5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} -0.3 & -0.7 \\ -0.1 & 0.3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-2} + \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t. \quad (2.10)$$

The process above is simulated in R code 2.1. This is achieved by employing the function `ARMA()` and its method `simulate()`, contained in the package `dse1` (see Gilbert [2004], [2000], [1995], and [1993]).² In the first step, the lag polynomial $A(L)$ as described in Equation (2.2) is created as an array signified by `Apoly`. The shape of the variance-covariance matrix of the error process is an identity matrix stored as object `B`, and finally the constant term is assigned as `TRD`. An `ARMA` object is created next, and the model is simulated for a sample size of 500 observations. The resultant series are retrieved from the list element `output` and plotted in Figure 2.1. In the next step, the lag order is empirically determined by utilizing `VARselect()`. Alternatively, the VAR(p)-model could have been estimated directly by setting `lag.max = 4` and `type = "AIC"`. All criteria indicate a lag order of two. Finally, a VAR(2) with a constant is estimated with function `VAR()`, and its roots are checked for stability by applying the function `roots()` to the object `varsimest`. 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.

The results of the VAR(2) for the variables y_1 and y_2 are presented in Tables 2.1 and 2.2, respectively. As expected, the estimated coefficients are close to their theoretical values, and all are significantly different from zero. Finally, the *eigenvalues* of the companion form are less than one and are provided in Table 2.3.

² Please note that this package is part of the bundle `dse`. As an alternative, a VAR-process can be simulated with the functions contained in the package `mAr`, too (see Barbosa [2007]).

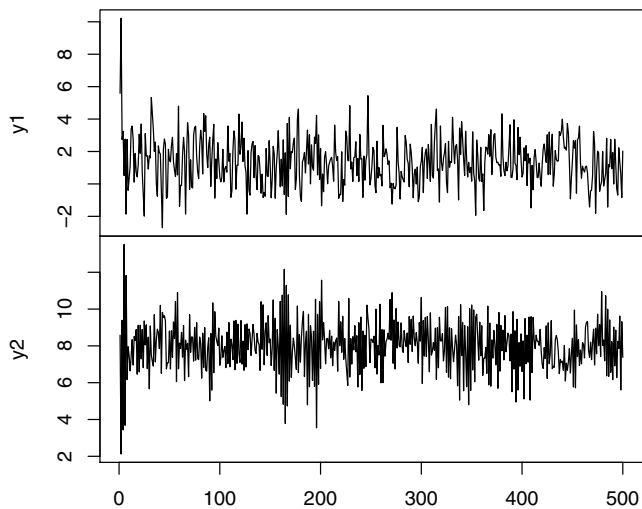


Fig. 2.1. Time series plot of the simulated VAR(2)-process

Table 2.2. VAR result for y_2

Variable	Estimate	Std. Error	t -value	$\Pr(> t)$
Lagged levels				
y1.l1	-0.1499	0.0358	-4.1920	0e + 00
y2.l1	-0.4740	0.0411	-11.5360	0e + 00
y1.l2	-0.1184	0.0355	-3.3328	9e - 04
y2.l2	0.3006	0.0458	6.5684	0e + 00
Deterministic				
const.	9.7620	0.6253	15.6124	0e + 00

Table 2.3. Eigenvalues of the companion form

	1	2	3	4
Eigenvalues	0.8311	0.6121	0.6121	0.6049

R Code 2.1 Simulation of VAR(2)-process

```

## Simulate VAR(2)-data 1
library(dse1) 2
library(vars) 3
## Setting the lag-polynomial A(L) 4
Apoly <- array(c(1.0, -0.5, 0.3, 0, 5
                 0.2, 0.1, 0, -0.2, 6
                 0.7, 1, 0.5, -0.3) , 7
               c(3, 2, 2)) 8
## Setting Covariance to identity-matrix 9
B <- diag(2) 10
## Setting constant term to 5 and 10 11
TRD <- c(5, 10) 12
## Generating the VAR(2) model 13
var2 <- ARMA(A = Apoly, B = B, TREND = TRD) 14
## Simulating 500 observations 15
varsim <- simulate(var2, sampleT = 500, 16
                  noise = list(w = matrix(rnorm(1000), 17
                                           nrow = 500, ncol = 2)), rng = list(seed = c(123456))) 18
## Obtaining the generated series 19
vardat <- matrix(varsim$output, nrow = 500, ncol = 2) 20
colnames(vardat) <- c("y1", "y2") 21
## Plotting the series 22
plot.ts(vardat, main = "", xlab = "") 23
## Determining an appropriate lag-order 24
infocrit <- VARselect(vardat, lag.max = 3, 25
                     type = "const") 26
## Estimating the model 27
varsimest <- VAR(vardat, p = 2, type = "const", 28
                season = NULL, exogen = NULL) 29
## Alternatively, selection according to AIC 30
varsimest <- VAR(vardat, type = "const", 31
                lag.max = 3, ic = "SC") 32
## Checking the roots 33
roots <- roots(varsimest) 34

```

2.2.2 Diagnostic Tests

Once a VAR-model has been estimated, it is of pivotal interest to see whether the residuals obey the model's assumptions. That is, one should check for the absence of serial correlation and heteroscedasticity and see if the error process is normally distributed. In Section 1.4, these kinds of tests were briefly introduced, and the versions will now be presented in more detail for the multivariate case. As a final check, one can conduct structural stability tests; *i.e.*, CUSUM, CUSUM-of-squares, and/or fluctuation tests. The latter tests can

be applied on a per-equation basis, whereas for the former tests multivariate statistics exist. All tests are made available in the package **vars**.

For testing the lack of serial correlation in the residuals of a VAR(p)-model, a Portmanteau test and the LM test proposed by Breusch [1978] and Godfrey [1978] are most commonly applied. For both tests, small sample modifications can be calculated, too, where the modification for the LM test was introduced by Edgerton and Shukur [1999]. The Portmanteau statistic is defined as

$$Q_h = T \sum_{j=1}^h \text{tr}(\hat{C}_j' \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) \quad (2.11)$$

with $\hat{C}_i = \frac{1}{T} \sum_{t=i+1}^T \hat{\mathbf{u}}_t \hat{\mathbf{u}}_t'$. 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)-model. The limiting distribution is only valid for h tending to infinity at a 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 chosen too large. The small-sample properties of the test statistic

$$Q_h^* = T^2 \sum_{j=1}^h \frac{1}{T-j} \text{tr}(\hat{C}_j' \hat{C}_0^{-1} \hat{C}_j \hat{C}_0^{-1}) \quad (2.12)$$

may be better.

The Breusch-Godfrey *LM*-statistic is based upon the following auxiliary regressions:

$$\hat{\mathbf{u}}_t = A_1 \mathbf{y}_{t-1} + \dots + A_p \mathbf{y}_{t-p} + C D_t + B_1 \hat{\mathbf{u}}_{t-1} + \dots + B_h \hat{\mathbf{u}}_{t-h} + \varepsilon_t. \quad (2.13)$$

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

$$LM_h = T(K - \text{tr}(\tilde{\Sigma}_R^{-1} \tilde{\Sigma}_e)), \quad (2.14)$$

where $\tilde{\Sigma}_R$ and $\tilde{\Sigma}_e$ assign the residual covariance matrix of the restricted and unrestricted models, respectively. The test statistic LM_h is distributed as $\chi^2(hK^2)$. Edgerton and Shukur [1999] proposed a small-sample correction, which is defined as

$$LMF_h = \frac{1 - (1 - R_r^2)^{1/r}}{(1 - R_r^2)^{1/r}} \frac{Nr - q}{Km}, \quad (2.15)$$

with $R_r^2 = 1 - |\tilde{\Sigma}_e|/|\tilde{\Sigma}_R|$, $r = ((K^2 m^2 - 4)/(K^2 + m^2 - 5))^{1/2}$, $q = 1/2 Km - 1$ and $N = T - K - m - 1/2(K - m + 1)$, where n is the number of regressors in the original system and $m = Kh$. The modified test statistic is distributed as $F(hK^2, \text{int}(Nr - q))$.

These tests are implemented in the function `serial.test()`. The test statistics are returned in the list element `serial` and have class attribute `htest`. Per default, the asymptotic Portmanteau test is returned. The adjusted version is computed if the `type` argument is set to `"PT.adjusted"`. The specifiers for the Breusch and Godfrey and the Edgerton and Shukur tests are `"BG"` and `"ES"`, respectively. The residuals are contained in the first list element. In R code 2.2, the asymptotic Portmanteau test is applied to the object `varsimest`.

R Code 2.2 Diagnostic tests of VAR(2)-process

```

## testing serial correlation                                1
args(serial.test)                                           2
## Portmanteau-Test                                         3
var2c.serial <- serial.test(varsimest, lags.pt = 16,        4
                             type = "PT.asymptotic")       5
var2c.serial                                                6
plot(var2c.serial, names = "y1")                            7
plot(var2c.serial, names = "y2")                            8
## testing heteroscedasticity                               9
args(arch.test)                                             10
var2c.arch <- arch.test(varsimest, lags.multi = 5,         11
                        multivariate.only = TRUE)          12
var2c.arch                                                  13
## testing for normality                                    14
args(normality.test)                                       15
var2c.norm <- normality.test(varsimest,                    16
                             multivariate.only = TRUE)    17
var2c.norm                                                  18
## class and methods for diganostic tests                  19
class(var2c.serial)                                         20
class(var2c.arch)                                           21
class(var2c.norm)                                           22
methods(class = "varcheck")                                 23
## Plot of objects "varcheck"                               24
args(vars:::plot.varcheck)                                 25
plot(var2c.serial, names = "y1")                           26

```

The implemented tests for heteroscedasticity are the univariate and multivariate ARCH tests (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 a special case of the exhibition below and is skipped):

$$vech(\hat{\mathbf{u}}_t \hat{\mathbf{u}}_t') = \beta_0 + B_1 vech(\hat{\mathbf{u}}_{t-1} \hat{\mathbf{u}}_{t-1}') + \dots + B_q vech(\hat{\mathbf{u}}_{t-q} \hat{\mathbf{u}}_{t-q}') + \mathbf{v}_t, \quad (2.16)$$

where \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 downward. The *vech* operation is easily applied to a matrix by using `lower.tri(..., diag = TRUE)`. The dimension of β_0 is $\frac{1}{2}K(K+1)$, and for the coefficient matrices B_i with $i = 1, \dots, q$, $\frac{1}{2}K(K+1) \times \frac{1}{2}K(K+1)$. The null hypothesis is $H_0 := B_1 = B_2 = \dots = B_q = 0$ and the alternative is $H_1 : B_1 \neq 0 \cap B_2 \neq 0 \cap \dots \cap B_q \neq 0$. The test statistic is defined as

$$\text{VARCH}_{LM}(q) = \frac{1}{2}TK(K+1)R_m^2, \quad (2.17)$$

with

$$R_m^2 = 1 - \frac{2}{K(K+1)}\text{tr}(\hat{\Omega}\hat{\Omega}_0^{-1}), \quad (2.18)$$

and $\hat{\Omega}$ assigns the covariance matrix of the regression model defined above. This test statistic is distributed as $\chi^2(qK^2(K+1)^2/4)$. These test statistics are implemented in the function `arch.test()` contained in the package **vars**. The default is to compute the multivariate test only. If `multivariate.only = FALSE`, the univariate tests are computed, too. In this case, the list object returned from `arch.test()` has three elements. The first element is the matrix of residuals. The second, signified by `arch.uni`, is a list object itself and holds the univariate test results for each of the series. The multivariate test result is contained in the third list element, signified by `arch.mul`. In R code 2.2, these tests are applied to the object `varsimest`.

The Jarque-Bera normality tests for univariate and multivariate series are implemented and applied to the residuals of a $\text{VAR}(p)$ as well as separate tests for multivariate skewness and kurtosis (see Bera and Jarque [1980], [1981], Jarque and 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 dependent upon the ordering of the variables. The test statistics for the multivariate case are defined as

$$JB_{mv} = s_3^2 + s_4^2, \quad (2.19)$$

where s_3^2 and s_4^2 are computed according to

$$s_3^2 = T\mathbf{b}'_1\mathbf{b}_1/6, \quad (2.20a)$$

$$s_4^2 = T(\mathbf{b}_2 - \mathbf{3}_K)'(\mathbf{b}_2 - \mathbf{3}_K)/24, \quad (2.20b)$$

and \mathbf{b}_1 and \mathbf{b}_2 are the third and fourth non-central moment vectors of the standardized residuals $\hat{\mathbf{u}}_t^s = \tilde{P}^-(\hat{\mathbf{u}}_t - \hat{\bar{\mathbf{u}}}_t)$ 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 $\chi^2(K)$.

These tests are implemented in the function `normality.test()` contained in the package **vars**. Please note that the default is to compute the multivariate tests only. To obtain the test statistics for the single residual series, the argument `multivariate.only` has to be set to `FALSE`. The list elements of this function returned are `jb.uni` and `jb.mul`, which consist of objects with class attribute `htest` as for the previously introduced tests.

The three former functions return a list object with class attribute `varcheck` for which `plot` and `print` methods exist. The plots — one for each equation — include a residual plot, an empirical distribution plot, and the ACF and PACF of the residuals and their squares. The `plot` method offers additional arguments for adjusting its appearance. The residual plots as returned by `plot(var2c.norm)`, for instance, are provided in Figures 2.2 and 2.3 for y_1 and y_2 , respectively. The results of the diagnostic tests are shown in Table 2.4.

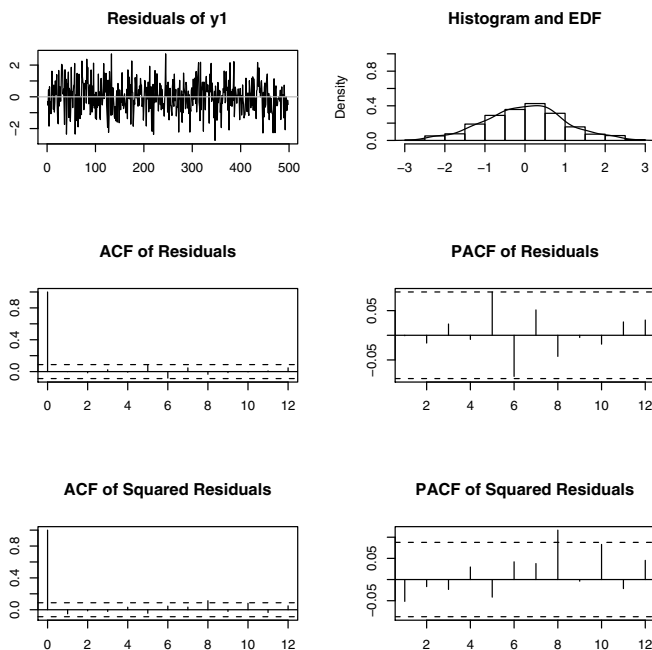


Fig. 2.2. Diagnostic residual plot for y_1 of VAR(2)-process

As expected for the simulated VAR(2)-process, none of the test outcomes indicate any deviations from a spherical error process.

Finally, structural stability can be tested by investigating the empirical fluctuation process. A detailed exposition of generalized fluctuation tests can be found for instance in Zeileis, Leisch, Hornik and Kleiber [2005] and Kuan

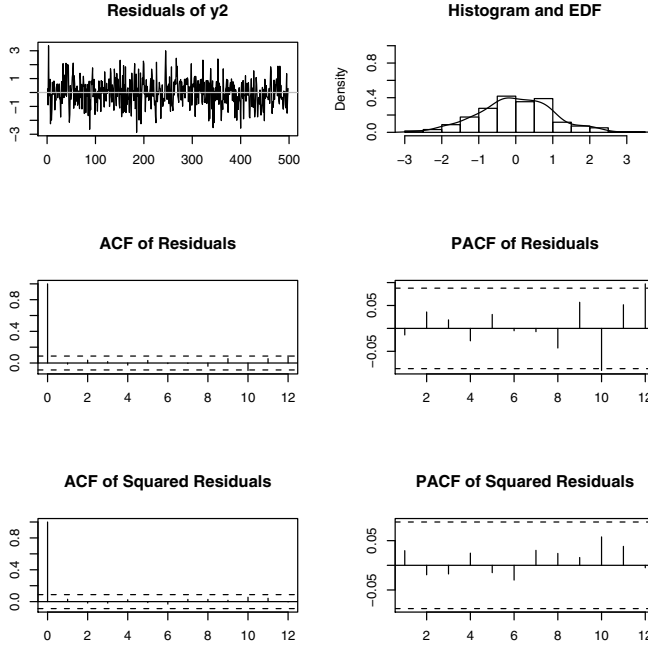


Fig. 2.3. Diagnostic residual plot for y_2 of VAR(2)-process

Table 2.4. Diagnostic tests of VAR(2)

Test	Statistic	D.F.	p -value
Portmanteau	52.44	56	0.61
ARCH VAR	32.58	45	0.92
JB VAR	0.54	4	0.97
Kurtosis	0.42	2	0.81
Skewness	0.12	2	0.94

and Hornik [1995]. Tests such as CUSUM, CUSUM-of-squares, MOSUM, and the fluctuation test are implemented in the function `efp()` contained in the package **strucchange**. The structural tests implemented in the package **strucchange** are explained in its vignette. The function `stability()` in the package **vars** is a wrapper function to `efp()`. The desired test is then applied to each of the equations in a VAR(p)-model. These kinds of tests are exhibited in R code 2.3, and their graphical results are depicted in Figures 2.4 and 2.5. In the code, an OLS-CUSUM and a fluctuation test have been applied to the simulated VAR(2)-process. In order to save space, only the test outcome for

y_1 (OLS-CUSUM test) and similarly the outcome for y_2 (fluctuation test) are shown. As expected, neither test indicates structural instability.

R Code 2.3 Empirical fluctuation processes

```
reccusum <- stability(varsimest ,                               1
  type = "OLS-CUSUM")                                         2
fluctuation <- stability(varsimest ,                             3
  type = "fluctuation")                                       4
```

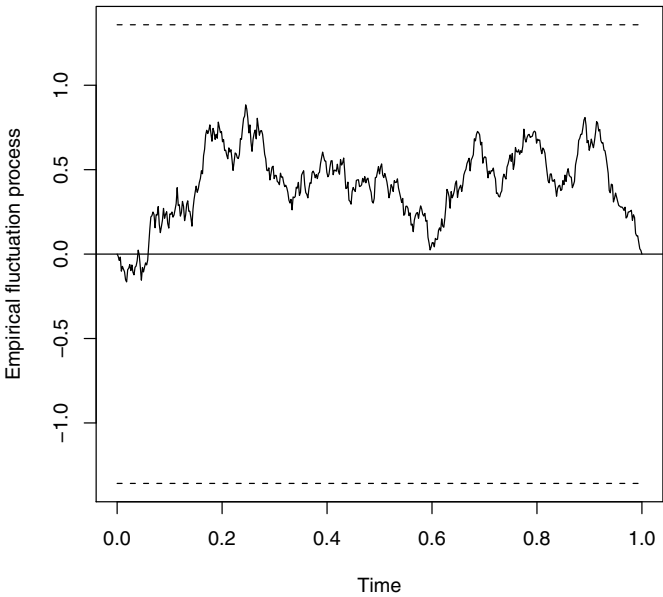


Fig. 2.4. OLS-CUSUM test for y_1 of VAR(2)-process

2.2.3 Causality Analysis

Often researchers are interested in the detection of causalities between variables. The most common one is the Granger causality test (see Granger [1969]). Incidentally, this test is not suited for testing causal relationships

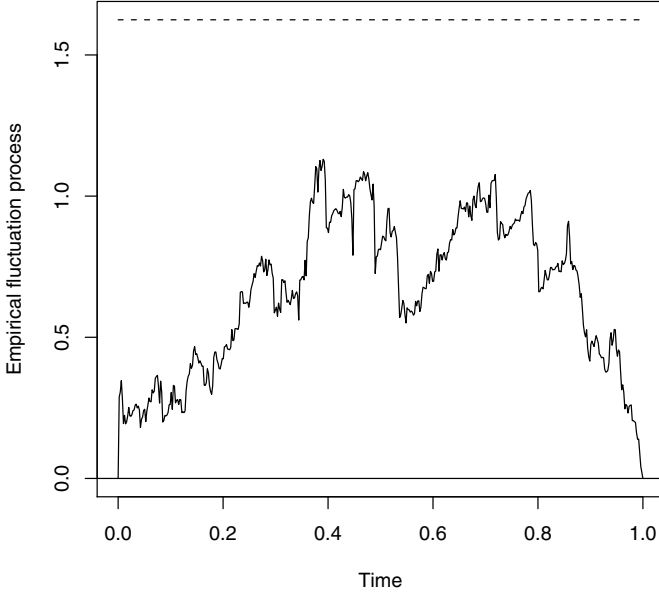


Fig. 2.5. Fluctuation test for y_2 of VAR(2)-process

in the strict sense because the possibility of a *post hoc ergo propter hoc* fallacy cannot be excluded. This is true for any “causality test” in econometrics. It is therefore common practice to say that variable x *granger-causes* variable y if variable x helps to predict variable y . Aside from this test, a Wald-type instantaneous causality test can be used, too. It is characterized by testing for non-zero correlation between the error processes of the cause and effect variables (see Lütkepohl [2006]).

For both tests, the vector of endogenous variables \mathbf{y}_t is split into two sub-vectors \mathbf{y}_{1t} and \mathbf{y}_{2t} with dimensions $(K_1 \times 1)$ and $(K_2 \times 1)$ with $K = K_1 + K_2$. For the rewritten VAR(p),

$$\begin{bmatrix} \mathbf{y}_{1t} \\ \mathbf{y}_{2t} \end{bmatrix} = \sum_{i=1}^p \begin{bmatrix} \alpha_{11,i} & \alpha_{12,i} \\ \alpha_{21,i} & \alpha_{22,i} \end{bmatrix} \begin{bmatrix} \mathbf{y}_{1,t-i} \\ \mathbf{y}_{2,t-i} \end{bmatrix} + CD_t + \begin{bmatrix} \mathbf{u}_{1t} \\ \mathbf{u}_{2t} \end{bmatrix}, \quad (2.21)$$

the null hypothesis that the sub-vector \mathbf{y}_{1t} does not Granger-cause \mathbf{y}_{2t} is defined as $\alpha_{21,i} = 0$ for $i = 1, 2, \dots, p$. The alternative is $\exists \alpha_{21,i} \neq 0$ for $i = 1, 2, \dots, p$. The test statistic is distributed as $F(pK_1K_2, KT - n^*)$, with n^* equal to the total number of parameters in the VAR(p)-process above, including deterministic regressors. The null hypothesis for non-instantaneous causality is defined as $H_0 : C\sigma = 0$, where C is an $(N \times K(K+1)/2)$ matrix

Table 2.5. Causality tests

Test	Statistic	<i>p</i> -value
Granger	250.07	0.00
Instant	0.00	0.99

of rank N selecting the relevant covariances of \mathbf{u}_{1t} and \mathbf{u}_{2t} ; $\tilde{\sigma} = \text{vech}(\tilde{\Sigma}_u)$. The Wald statistic is defined as

$$\lambda_W = T\tilde{\sigma}'C'[2CD_K^+(\tilde{\Sigma}_u \otimes \tilde{\Sigma}_u)D_K^{+'}C']^{-1}C\tilde{\sigma}, \quad (2.22)$$

where the Moore-Penrose inverse of the duplication matrix D_K is assigned by 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 , $\text{vec}(A) = D_K\text{vech}(A)$ holds. The test statistic λ_W is asymptotically distributed as $\chi^2(N)$.

Both tests are implemented in the function `causality()` contained in the package `vars`. The function has two arguments. The first argument, `x`, is an object of class `varrest`, and the second, `cause`, is a character vector of the variable names, which 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 the cause variable and a warning is printed. In R code 2.4, this function is applied to the simulated VAR(2)-process. The results are provided in Table 2.5. Clearly, the null hypothesis of no Granger causality has to be dismissed, whereas the hypothesis of no instantaneous causality cannot be rejected.

R Code 2.4 Causality analysis of VAR(2)-process

```
## Causality tests 1
## Granger and instantaneous causality 2
var.causal <- causality(varsimest, cause = "y2") 3
```

2.2.4 Forecasting

Once a VAR-model has been estimated and passes the diagnostic tests, it can be used for forecasting. Indeed, one of the primary purposes of VAR analysis is the detection of the dynamic interaction between the variables included in a VAR(p)-model. Aside from forecasts, other tools for investigating these relationships are impulse response analysis and forecast error variance decomposition, which will be covered in Subsections 2.2.5 and 2.2.6, respectively.

For a given empirical VAR, forecasts can be calculated recursively according to

$$\mathbf{y}_{T+h|T} = A_1 \mathbf{y}_{T+h-1|T} + \dots + A_p \mathbf{y}_{T+h-p|T} + CD_{T+h} \quad (2.23)$$

for $h = 1, 2, \dots, n$. The forecast error covariance matrix is given as

$$\text{Cov} \left(\begin{bmatrix} \mathbf{y}_{T+1} - \mathbf{y}_{T+1|T} \\ \vdots \\ \mathbf{y}_{T+h} - \mathbf{y}_{T+h|T} \end{bmatrix} \right) = \begin{bmatrix} I & 0 & \dots & 0 \\ \Phi_1 & I & & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \dots & I \end{bmatrix} (\Sigma_{\mathbf{u}} \otimes I_h)$$

$$\begin{bmatrix} I & 0 & \dots & 0 \\ \Phi_1 & I & & 0 \\ \vdots & & \ddots & 0 \\ \Phi_{h-1} & \Phi_{h-2} & \dots & I \end{bmatrix}',$$

and the matrices Φ_i are the coefficient matrices of the Wold moving average representation of a stable VAR(p)-process. Forecast confidence bands can then be calculated as

$$[y_{k,T+h|T} - c_{1-\gamma/2}\sigma_k(h), y_{k,T+h|T} + c_{1-\gamma/2}\sigma_k(h)], \quad (2.24)$$

where $c_{1-\gamma/2}$ signifies the $(1 - \frac{\gamma}{2})$ percentage point of the normal distribution and $\sigma_k(h)$ is the standard deviation of the k th variable h steps ahead.

In the package **vars**, forecasting of VAR-processes is accomplished by a **predict** method for objects with class attribute **varest**. Besides 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 **predict** method returns 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 chosen 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** exists as well as a **fanchart()** function for plotting *fan charts* as described in Britton, Fisher and Whitley [1998].

In R code 2.5, the **predict** method is applied to the empirical simulated VAR-process. 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 gray color scheme is used and the critical values are set from 0.1 to 0.9 with a step size of 0.1. The predictions for y_1 are shown in Figure 2.6, and the fan chart for variable y_2 is depicted in Figure 2.7.

2.2.5 Impulse Response Functions

In Subsection 2.2.3, two causality tests were introduced, that are quite useful to infer whether a variable helps predict another one. However, this analysis

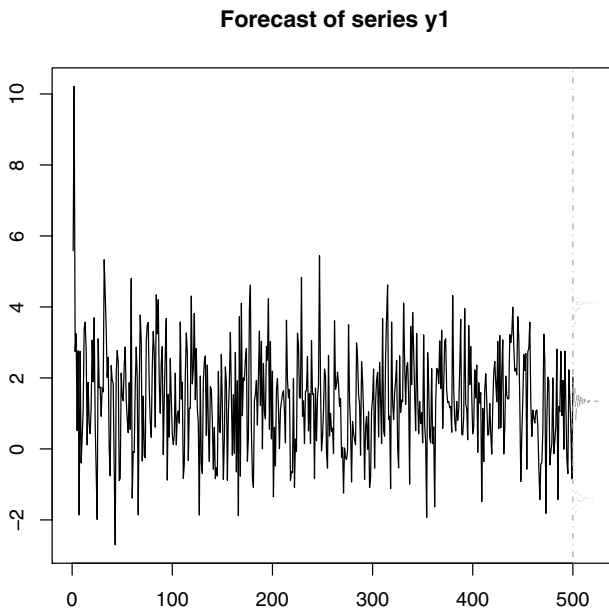
R Code 2.5 Forecasts of VAR-process

```

## Forecasting objects of class varest                                1
args(vars:::predict.varest)                                          2
predictions <- predict(varsimest, n.ahead = 25,                      3
                      ci = 0.95)                                     4

class(predictions)                                                  5
args(vars:::plot.varprd)                                            6
## Plot of predictions for y1                                       7
plot(predictions, names = "y1")                                     8
## Fanchart for y2                                                  9
args(fanchart)                                                      10
fanchart(predictions, names = "y2")                                 11

```

**Fig. 2.6.** Forecasting y_1 of VAR(2)-process

falls short of quantifying the impact of the impulse variable on the response variable over time. The *impulse response analysis* is used to investigate these kinds of dynamic interactions between the endogenous variables and is based upon the Wold moving average representation of a VAR(p)-process (see Equations (2.6) and (2.7)). The (i, j) th coefficients of the matrices Φ_s are thereby

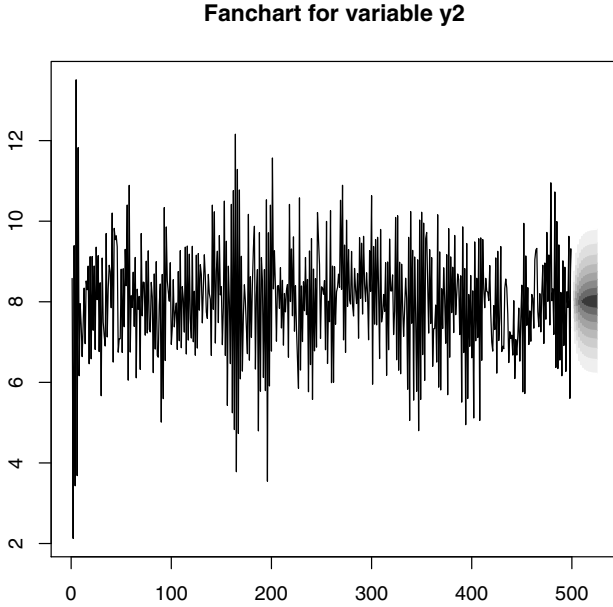


Fig. 2.7. Fanchart of y_2 of VAR(2)-process

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 $s = 1, 2, \dots$, and hence one would obtain the cumulated impact of a unit change in variable j on the variable i at time s . Rather than these impulse response coefficients, it is often conceivable to use orthogonal impulse responses 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 exists; *i.e.*, the off-diagonal elements of $\Sigma_{\mathbf{u}}$ are non-zero. The orthogonal impulse responses are derived from a Choleski decomposition of the error variance-covariance matrix $\Sigma_{\mathbf{u}} = PP'$, with P being lower triangular. The moving average representation can then be transformed to

$$\mathbf{y}_t = \Psi_0 \varepsilon_t + \Psi_1 \varepsilon_{t-1} + \dots, \quad (2.25)$$

with $\varepsilon_t = P^{-1}\mathbf{u}_t$ and $\Psi_i = \Phi_i P$ for $i = 0, 1, 2, \dots$ 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 exerts an influence on all the remaining ones and that the second and following variables cannot have a direct impact on y_{1t} . Hence, a certain structure of the error terms is implicitly imposed. One should bear this in mind when orthogonal impulse responses are employed.

Please note further that a different ordering of the variables might produce different outcomes with respect to the impulse responses. As we shall see in Section 2.3, the non-uniqueness of the impulse responses can be circumvented by analyzing a set of endogenous variables in the SVAR framework.

The function for conducting impulse response analysis is `irf()`, contained in the package `vars`. It is a method for objects with class attribute `varest`. 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 responses is set to 10 via argument `n.ahead`. The computation of orthogonal 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 replicable 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 bootstrapped impulse response coefficients $\hat{\Phi}^*$ or $\hat{\Psi}^*$ (see Efron and Tibshirani [1993]). The function `irf()` returns an object with class attribute `varirf` for which a `plot` and a `print` method exist.

In R code 2.6, an impulse response analysis is conducted for the simulated VAR(2)-process. For clarity, the impulse responses of y_1 to y_2 and vice versa have been split into two separate command lines. The results are shown in Figures 2.8 and 2.9, respectively.

R Code 2.6 IRA of VAR-process

```
## Impulse response analysis
irf.y1 <- irf(varsimest, impulse = "y1",
              response = "y2", n.ahead = 10,
              ortho = FALSE, cumulative = FALSE,
              boot = FALSE, seed = 12345)
args(vars::plot.varirf)
plot(irf.y1)
irf.y2 <- irf(varsimest, impulse = "y2",
              response = "y1", n.ahead = 10,
              ortho = TRUE, cumulative = TRUE,
              boot = FALSE, seed = 12345)
plot(irf.y2)
```

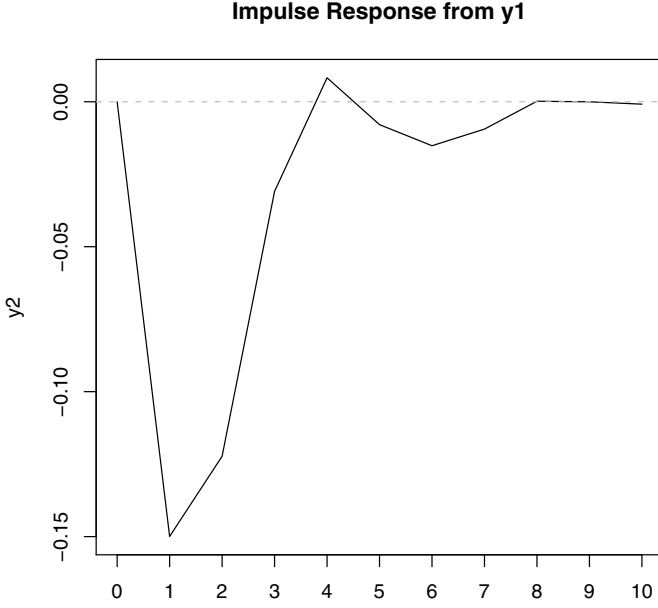


Fig. 2.8. Impulse responses of y_1 to y_2

2.2.6 Forecast Error Variance Decomposition

The forecast error variance decomposition (FEVD) is based upon the orthogonal impulse response coefficient matrices Ψ_n (see Subsection 2.2.4). The FEVD allows the user to analyze the contribution of variable j to the h -step forecast error variance of variable k . If the element-wise squared orthogonal impulse responses are divided by the variance of the forecast error variance, $\sigma_k^2(h)$, the result is a percentage figure. Formally, the forecast error variance for $y_{k,T+h} - Y_{k,T+h|T}$ is defined as

$$\sigma_k^2(h) = \sum_{n=0}^{h-1} (\psi_{k1,n}^2 + \dots + \psi_{kK,n}^2), \quad (2.26)$$

which can be written as

$$\sigma_k^2(h) = \sum_{j=1}^K (\psi_{kj,0}^2 + \dots + \psi_{kj,h-1}^2). \quad (2.27)$$

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

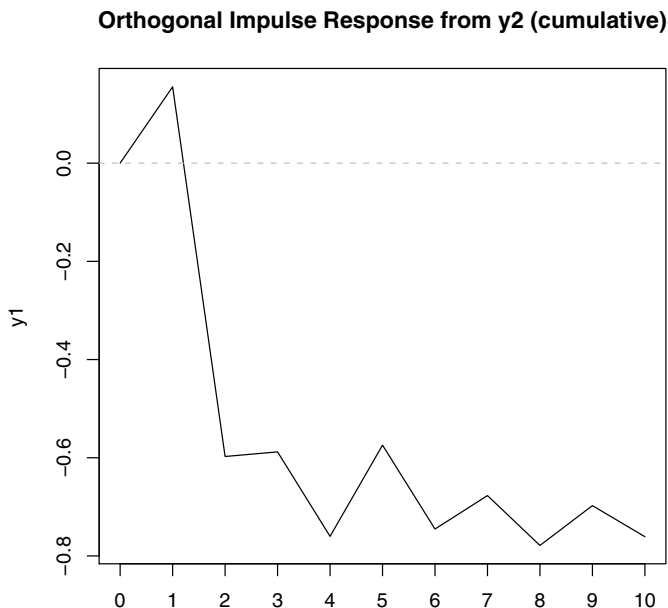


Fig. 2.9. Impulse responses of y_2 to y_1

$$\omega_{kj}(h) = (\psi_{kj,0}^2 + \dots + \psi_{kj,h-1}^2) / \sigma_k^2(h). \quad (2.28)$$

The `fevd` method in the package **vars** is available for conducting FEVD. The argument `n.ahead` sets the number of forecasting steps; it has a default value of 10. In R code 2.7, an FEVD is applied to the simulated VAR(2)-process, and its graphical output is presented in Figure 2.10.

R Code 2.7 FEVD of VAR-process

<code>## Forecast error variance decomposition</code>	1
<code>fevd.var2 <- fevd(varsimest, n.ahead = 10)</code>	2
<code>args(vars:::plot.varfevd)</code>	3
<code>plot(fevd.var2, addbars = 2)</code>	4

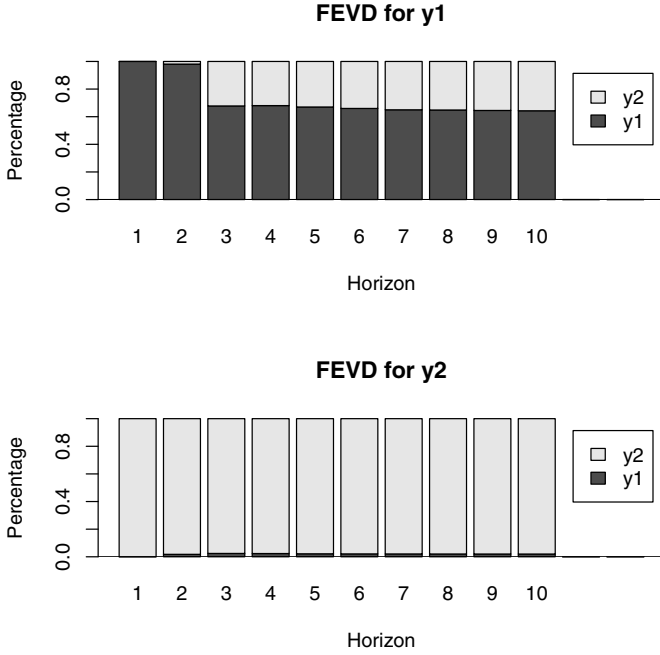


Fig. 2.10. FEVD for VAR(2)-process

2.3 Structural Vector Autoregressive Models

2.3.1 Specification and Assumptions

Recall from Subsection 2.2.1 the definition of a VAR(p)-process, in particular Equation (2.1). A VAR(p) can be interpreted as a reduced-form model. An SVAR model is its structural form and is defined as

$$A\mathbf{y}_t = A_1^*\mathbf{y}_{t-1} + \dots + A_p^*\mathbf{y}_{t-p} + B\varepsilon_t. \quad (2.29)$$

For a textbook exposition of SVAR-models, see Amisano and Giannini [1997]. It is assumed that the *structural errors*, ε_t , are white noise and the coefficient matrices A_i^* for $i = 1, \dots, p$, are structural coefficients that will differ from their reduced-form counterparts if $A \neq I$. To see this, consider the resulting equation by left-multiplying Equation (2.29) with the inverse of A :

$$\begin{aligned} \mathbf{y}_t &= A^{-1}A_1^*\mathbf{y}_{t-1} + \dots + A^{-1}A_p^*\mathbf{y}_{t-p} + A^{-1}B\varepsilon_t, \\ \mathbf{y}_t &= A_1\mathbf{y}_{t-1} + \dots + A_p\mathbf{y}_{t-p} + \mathbf{u}_t. \end{aligned} \quad (2.30)$$

An 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 an 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 an 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 restrictions imposed, three types of SVAR-models can be distinguished:

- A-model: B is set to I_K (minimum number of restrictions for identification is $K(K-1)/2$).
- B-model: A is 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 $K^2 + K(K-1)/2$).

2.3.2 Estimation

Depending on the SVAR type, the estimation is similar to the estimation of a simultaneous multiple-equation model with covariance restrictions on the error terms. In practice, the maximum-likelihood principle is applied to the concentrated log-likelihood, which is given as

$$\ln L_c(A, B) = \text{const} + \frac{T}{2} \ln |A^2| - \frac{T}{2} \ln |B^2| - \frac{T}{2} \text{tr}(A'B'^{-1}B^{-1}A\hat{\Sigma}_u), \quad (2.31)$$

where $\hat{\Sigma}_u$ signifies the estimated residual covariance matrix of the VAR(p)-model. The negative of Equation (2.31) is minimized subject to the imposed restrictions on A and B , which can be compactly written as

$$\begin{bmatrix} \text{vec} A \\ \text{vec} B \end{bmatrix} = \begin{bmatrix} R_A & 0 \\ 0 & R_B \end{bmatrix} \begin{bmatrix} \gamma_A \\ \gamma_b \end{bmatrix} + \begin{bmatrix} r_A \\ r_B \end{bmatrix}. \quad (2.32)$$

Two approaches for numerically estimating the unknown coefficients are implemented within the R package **vars**. The first method applies the `optim()` function for direct minimization of the negative log-likelihood, whereas the second method makes use of the scoring algorithm proposed by Amisano and Giannini [1997]. Either method is selected by providing "direct" or "scoring" as the value for the argument `estmethod` in the function `SVAR()`. In addition, the first argument in a call to `SVAR()` must be an object of class `varest`. 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 is 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` is 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. Depending on the chosen model, the list elements **A**, **Ase**, **B**, **Bse** contain the estimated coefficient matrices with the numerical standard errors, if applicable. In case `estmethod = "direct"`, the standard errors are returned only if `SVAR()` has been called with `hessian = TRUE`. The returned list element **Sigma.U** is the variance-covariance matrix of the reduced-form residuals times 100; *i.e.*, $\Sigma_U = A^{-1}BB'A^{-1'} \times 100$. Please note that this estimated variance-covariance matrix only corresponds to the reduced-form counterpart if the SVAR-model is exactly identified. The validity of the overidentifying restrictions can be tested with an LR test defined as

$$LR = T(\ln |\tilde{\Sigma}_u| - \ln |\hat{\Sigma}_u|), \quad (2.33)$$

where $\tilde{\Sigma}_u$ is the implied variance-covariance matrix of the SVAR and $\hat{\Sigma}_u$ signifies the reduced-form counterpart. The statistic is distributed as χ^2 with degrees of freedom equal to the number of overidentifying restrictions. This test statistic is returned as list element **LR** with class attribute **htest**. The element **opt** is the object returned 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()`.

In R code 2.8, the function `SVAR()` is applied to a generated A-model of the form

$$\begin{bmatrix} 1 & -0.7 \\ 0.8 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_t = \begin{bmatrix} 0.5 & 0.2 \\ -0.2 & -0.5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-1} + \begin{bmatrix} -0.3 & -0.7 \\ -0.1 & 0.3 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}_{t-2} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \end{bmatrix}_t. \quad (2.34)$$

In the call to `SVAR()`, the argument `hessian = TRUE` has been used, which is passed to `optim()`. Hence, the empirical standard errors are returned in the list element **Ase** of the object `svar.A`. The result is shown in Table 2.6. The coefficients are close to their theoretical counterparts and statistically significant different from zero. As expected, the likelihood-ratio statistic for overidentification does not indicate the rejection of the null.

Table 2.6. SVAR A-model: estimated coefficients

Variable	y_1	y_2
y_1	1.0000	-0.6975 (-13.67)
y_2	0.8571 (14.96)	1.0000

Note: t statistics in parentheses.

R Code 2.8 SVAR: A-model

```

library(dse1) 1
library(vars) 2
## A-model 3
Apoly <- array(c(1.0, -0.5, 0.3, 0.8, 4
                 0.2, 0.1, -0.7, -0.2, 5
                 0.7, 1, 0.5, -0.3) , 6
               c(3, 2, 2)) 7
## Setting covariance to identity-matrix 8
B <- diag(2) 9
## Generating the VAR(2) model 10
svarA <- ARMA(A = Apoly, B = B) 11
## Simulating 500 observations 12
svarsim <- simulate(svarA, sampleT = 500, 13
                   rng = list(seed = c(123456))) 14
## Obtaining the generated series 15
svardat <- matrix(svarsim$output, nrow = 500, ncol = 2) 16
colnames(svardat) <- c("y1", "y2") 17
## Estimating the VAR 18
varest <- VAR(svardat, p = 2, type = "none") 19
## Setting up matrices for A-model 20
Amat <- diag(2) 21
Amat[2, 1] <- NA 22
Amat[1, 2] <- NA 23
## Estimating the SVAR A-type by direct maximisation 24
## of the log-likelihood 25
args(SVAR) 26
svar.A <- SVAR(varest, estmethod = "direct", 27
               Amat = Amat, hessian = TRUE) 28

```

A B-type SVAR is first simulated and then estimated with the alternative method `estmethod = "scoring"` in R code 2.9. The scoring algorithm is based upon the updating equation

$$\begin{bmatrix} \tilde{\gamma}_A \\ \tilde{\gamma}_B \end{bmatrix}_{i+1} = \begin{bmatrix} \tilde{\gamma}_A \\ \tilde{\gamma}_B \end{bmatrix}_i + \ell \mathcal{I} \left(\begin{bmatrix} \tilde{\gamma}_A \\ \tilde{\gamma}_B \end{bmatrix}_i \right)^{-1} \mathcal{S} \left(\begin{bmatrix} \tilde{\gamma}_A \\ \tilde{\gamma}_B \end{bmatrix}_i \right), \quad (2.35)$$

where ℓ signifies the step length, \mathcal{I} is the information matrix for the unknown coefficients contained in $\tilde{\gamma}_A$ and $\tilde{\gamma}_B$, and \mathcal{S} is the scoring vector. The iteration step is assigned by i .

The covariance for the error terms has been set to -0.8 . The values of the coefficient matrices A_i for $i = 1, 2$ are the same as in the previous example. The result is provided in Table 2.7.

In addition to an object with class attribute `varest`, the other arguments of `SVAR()` if `estmethod = "scoring"` are `max.iter` for defining the maximal number of iterations, `conv.crit` for providing a value for defining convergence,

R Code 2.9 SVAR: B-model

```

library(dse1)                                1
library(vars)                                2
## B-model                                    3
Apoly <- array(c(1.0, -0.5, 0.3, 0,          4
                 0.2, 0.1, 0, -0.2,          5
                 0.7, 1, 0.5, -0.3) ,        6
               c(3, 2, 2))                   7
## Setting covariance to identity-matrix      8
B <- diag(2)                                  9
B[2, 1] <- -0.8                               10
## Generating the VAR(2) model                11
svarB <- ARMA(A = Apoly, B = B)               12
## Simulating 500 observations                13
svarsim <- simulate(svarB, sampleT = 500,      14
                   rng = list(seed = c(123456))) 15
svardat <- matrix(svarsim$output, nrow = 500, ncol = 2) 16
colnames(svardat) <- c("y1", "y2")           17
varest <- VAR(svardat, p = 2, type = "none")   18
## Estimating the SVAR B-type by scoring algorithm 19
## Setting up the restriction matrix and vector 20
## for B-model                               21
Bmat <- diag(2)                               22
Bmat[2, 1] <- NA                              23
svar.B <- SVAR(varest, estmethod = "scoring", 24
              Bmat = Bmat, max.iter = 200)     25

```

Table 2.7. SVAR B-model: estimated coefficients

Variable	y_1	y_2
y_1	1.0000	0.0000
y_2	-0.8439 (-18.83)	1.0000

Note: t statistics in parentheses.

and `maxls` for determining the maximal step length. As in the estimation method `direct`, the alternative method returns an object with class attribute `svarest`. For objects of this class, methods for computing impulse responses and forecast error variance decomposition exist. These methods will be the subjects of the following two subsections.

2.3.3 Impulse Response Functions

Just as impulse response analysis can be conducted for objects with class attribute `varest`, it can also be done for objects with class attribute `svarest`

(see Subsection 2.2.5 on page 37 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 an SVAR. The impulse response coefficients for an SVAR are calculated as $\Theta_i = \Phi_i A^{-1} B$ for $i = 1, \dots, n$.

In R code 2.10, IRA is exhibited for the estimated A-type SVAR from the previous section. The impulses from `y1` to `y2` are calculated. In program line 3, the method is applied to the object `svar.A`. In line 6, these orthogonal impulse responses are plotted. The result is provided in Figure 2.11.

R Code 2.10 SVAR: Impulse response analysis

```
## Impulse response analysis of SVAR A-type model      1
args(vars:::irf.svarest)                             2
irf.svara <- irf(svar.A, impulse = "y1",              3
                 response = "y2", boot = FALSE)       4
args(vars:::plot.varirf)                             5
plot(irf.svara)                                       6
```

2.3.4 Forecast Error Variance Decomposition

A forecast error variance decomposition can be applied to objects of class `svarest`. Here the forecast errors of $y_{T+h|T}$ are derived from the impulse responses of an SVAR, and the derivation for the forecast error variance decomposition is similar to the one outlined for the VAR-model (see Subsection 2.2.6 on page 41 following).

R Code 2.11 SVAR: Forecast error variance decomposition

```
## FEVD analysis of SVAR B-type model                1
args(vars:::fevd.svarest)                             2
fevd.svarb <- fevd(svar.B, n.ahead = 5)               3
class(fevd.svarb)                                       4
methods(class = "varfevd")                             5
plot(fevd.svarb)                                       6
```

An application for the SVAR B-model is provided in R code 2.11. As for the FEVD for VAR-models, `print` and `plot` methods exist for SVAR-models. The outcome of the FEVD for the variable `y2` is provided in Table 2.8.

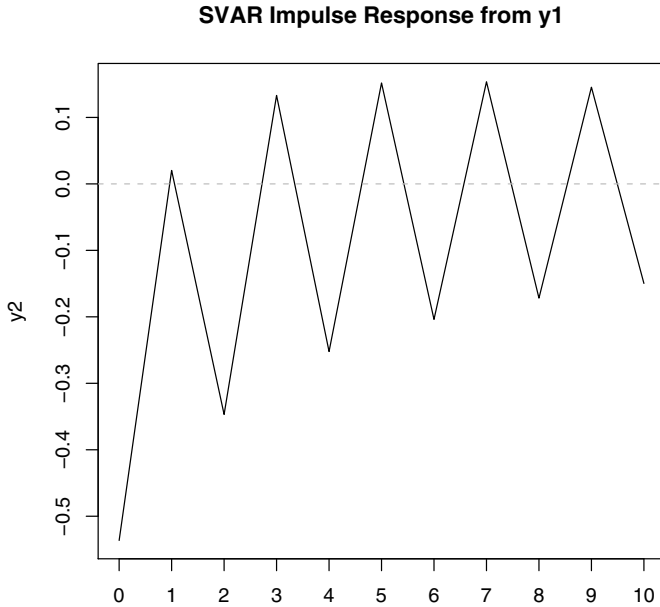


Fig. 2.11. IRA from y_1 to y_2 of SVAR A-model

Table 2.8. SVAR B-model: FEVD for y_2

Period	y_1	y_2
1	0.4160	0.5840
2	0.4021	0.5979
3	0.4385	0.5615
4	0.4342	0.5658
5	0.4350	0.5650

Summary

In this chapter, the analysis of stationary time series has been extended to multivariate models and their associated statistical tests and methods. In particular, VAR- and SVAR-models have been introduced, where the former can be interpreted as the reduced-form counterparts of SVAR-models. Both model classes have been illustrated by artificial data sets.

It has been outlined how a suitable lag length can be empirically determined and what kind of diagnostic tests are at hand for checking the assumptions about the multivariate error process. The different concepts of causality

analysis and forecasting with VAR-models have been shown. For investigating the dynamic interactions between variables, the impulse response functions and forecast error variance decomposition have been introduced. These tools are implemented as methods for VAR- and SVAR-models alike. The results can be obtained and plotted swiftly with the functions included in package **vars**. An overview of the package’s structure is presented in Table 2.9.

Exercises

- 1. Set up a three-dimensional VAR(2) model where the third variable does not Granger-cause the first variable.
- 2. Simulate 250 observations of your model from Exercise 1.
- 3. Estimate a VAR(2)-model with the simulated data from Exercise 2 and check its stability.
- 4. Conduct the diagnostic tests outlined in Subsection 2.2.2.
- 5. Perform Granger-causality tests for y_3 , Granger-causing y_2 and y_1 .
- 6. Calculate the impulse response functions (orthogonal and non-orthogonal) and forecast error variance decomposition for y_3 .

Table 2.9. Overview of package **vars**

function or method class		methods for class	functions for class
VAR	varest	coef, fevd, fitted, irf, logLik, Phi, plot, predict, print, Psi, resid, summary	Acoef, arch.test, Bcoef, BQ, causality, normality.test, restrict, roots, serial.test, stability
SVAR	svarest	fevd, irf, logLik, Phi, print, summary	
SVEC	svecest	fevd, irf, logLik, Phi, print, summary	
vec2var	vec2var	fevd, fitted, irf, logLik, Phi, predict, print, Psi, resid	arch.test, normality.test, serial.test
fevd	varfevd	plot, print	
irf	varirf	plot, print	
predict	varprd	plot, print	fanchart
summary	varsum, svarsum, svecsum	print	
arch.test	varcheck	plot, print	
normality.test	varcheck	plot, print	
serial.test	varcheck	plot, print	
stability	varstabil	plot, print	

7. Set up an SVAR-model of type AB with three variables and two lags, which are just identified and overidentified, respectively.
8. Simulate 250 observations of your model from Exercise 7 and estimate it with function **SVAR2**.
9. Perform impulse response analysis and forecast error variance decomposition of your estimated SVAR AB-model.

Analysis of Integrated and Cointegrated Time Series
with R

Pfaff, B.

2008, XX, 190 p., Softcover

ISBN: 978-0-387-75966-1