

Winter Term 2019/2020

Dr. Yannick Hoga Thilo Reinschlüssel

Multivariate Time Series Analysis

Exercise Sheet 2

1 Exercise 1: Moments and Simulation of a VAR(1) Process

Take the model from Example 2.4 on Slide 2-6:

- a) Derive a formula to obtain the population cross-covariance matrices for the lags 1 to 10 and compute them using R.

Hint: A glance at the slides and a loop might save you some time.

- b) Based on your results, compute the cross-correlation matrices.

Solution:

$$z_t = \phi_1 z_{t-1} + a_t$$

$$\begin{aligned}\Gamma_0 &= \phi_1 \Gamma_0 \phi_1' + \Sigma_a \\ \Leftrightarrow \text{vec}(\Gamma_0) &= (\phi_1 \otimes \phi_1) \cdot \text{vec}(\Gamma_0) + \text{vec}(\Sigma_a) \\ \Leftrightarrow \text{vec}(\Gamma_0) &= (I_{K^2} - \phi_1 \otimes \phi_1) \cdot \text{vec}(\Sigma_a) \\ \Rightarrow \Gamma_1 &= \phi_1 \Gamma_0 \\ \Rightarrow \Gamma_l &= \phi_{l-1} \Gamma_{l-1} = \phi_1^l \Gamma_0\end{aligned}$$

First define parameters/coefficients:

```
Phi <- matrix(data = c(0.2, -0.6, 0.3, 1.1),  
              byrow = FALSE, nrow = 2) # VAR coefficients
```

```

k <- ncol(Phi)
Sigma_a <- matrix(data = c(1, 0.8, 0.8, 2.0),
                  byrow = FALSE, nrow = 2) # innovations' covariances

Phi_kron <- kronecker(X = Phi, Y = Phi) # kronecker product
# Phi %% Phi # alternative command

Ident <- diag(ncol(Phi_kron))
# identity matrix with the same dimensions as Phi_kron

Gamma0.vec <- solve(Ident - Phi_kron) %% c(Sigma_a)
# c() works like the "vec" operator

Gamma0.mat <- matrix(data = Gamma0.vec, nrow = 2, byrow = FALSE)

```

The Γ_0 matrix is than:

```

##          [,1]      [,2]
## [1,] 2.288889 3.511111
## [2,] 3.511111 8.622222

```

To derive the Γ_1 matrix, we can apply the following formula:

$$\Gamma_1 = \phi_1 \Gamma_0$$

```
Gamma1.mat <- Phi %% Gamma0.mat
```

Γ_1 is than:

```

##          [,1]      [,2]
## [1,] 1.511111 3.288889
## [2,] 2.488889 7.377778

```

To derive all desired l^{Th} lagged covariance matrices we can use the general equation and program a loop over all desired lags:

$$\Rightarrow \Gamma_l = \phi_{l-1} \Gamma_{l-1} = \phi_1^l \Gamma_0$$

To derive the correlation matrices ρ_l we divide the covariances by the standard deviations:

$$\rho_l = \begin{pmatrix} \frac{\text{Cov}(x_t, x_{t-l})}{\sqrt{\text{Var}(x_t) \cdot \text{Var}(x_{t-l})}} & \frac{\text{cov}(x_t, y_{t-l})}{\sqrt{\text{Var}(x_t) \cdot \text{Var}(y_{t-l})}} \\ \frac{\text{cov}(y_t, x_{t-l})}{\sqrt{\text{Var}(y_t) \cdot \text{Var}(x_{t-l})}} & \frac{\text{cov}(y_t, y_{t-l})}{\sqrt{\text{Var}(y_t) \cdot \text{Var}(y_{t-l})}} \end{pmatrix}$$

```
# compute further (lagged) cross-covariance-functions
# preparing the variables to store the matrices into: covariance
ccovf.list <- list()
# "" : correlation
ccorf.list <- list()

# obtaining standard deviations for the single variables (parts of z)
#and putting them in a diagonal matrix
D.inv <- solve(sqrt(Gamma0.mat * diag(ncol(Phi))))
for (i in 1:10){
  ccovf.list[[i]] <- Phi%^i %*% Gamma0.mat
  ccorf.list[[i]] <- D.inv %*% ccovf.list[[i]] %*% D.inv
}
ccovf.list # cross lagged covariances
```

```
## [[1]]
##           [,1]      [,2]
## [1,] 1.511111 3.288889
## [2,] 2.488889 7.377778
##
## [[2]]
##           [,1]      [,2]
## [1,] 1.048889 2.871111
## [2,] 1.831111 6.142222
##
## [[3]]
```

```

##          [,1]      [,2]
## [1,] 0.7591111 2.416889
## [2,] 1.3848889 5.033778
##
## [[4]]
##          [,1]      [,2]
## [1,] 0.5672889 1.993511
## [2,] 1.0679111 4.087022
##
## [[5]]
##          [,1]      [,2]
## [1,] 0.4338311 1.624809
## [2,] 0.8343289 3.299618
##
## [[6]]
##          [,1]      [,2]
## [1,] 0.3370649 1.314847
## [2,] 0.6574631 2.654694
##
## [[7]]
##          [,1]      [,2]
## [1,] 0.2646519 1.059378
## [2,] 0.5209705 2.131255
##
## [[8]]
##          [,1]      [,2]
## [1,] 0.2092215 0.8512522
## [2,] 0.4142764 1.7087543
##
## [[9]]
##          [,1]      [,2]
## [1,] 0.1661272 0.6828767
## [2,] 0.3301711 1.3688784
##
## [[10]]
##          [,1]      [,2]
## [1,] 0.1322768 0.5472389
## [2,] 0.2635119 1.0960403

```

```
ccorf.list # cross lagged correlations
```

```
## [[1]]
##           [,1]      [,2]
## [1,] 0.6601942 0.7403332
## [2,] 0.5602522 0.8556701
##
## [[2]]
##           [,1]      [,2]
## [1,] 0.4582524 0.6462909
## [2,] 0.4121855 0.7123711
##
## [[3]]
##           [,1]      [,2]
## [1,] 0.3316505 0.5440449
## [2,] 0.3117403 0.5838144
##
## [[4]]
##           [,1]      [,2]
## [1,] 0.2478447 0.4487420
## [2,] 0.2403882 0.4740103
##
## [[5]]
##           [,1]      [,2]
## [1,] 0.1895379 0.3657466
## [2,] 0.1878085 0.3826876
##
## [[6]]
##           [,1]      [,2]
## [1,] 0.1472614 0.2959738
## [2,] 0.1479958 0.3078898
##
## [[7]]
##           [,1]      [,2]
## [1,] 0.1156246 0.2384673
## [2,] 0.1172711 0.2471817
##
## [[8]]
##           [,1]      [,2]
## [1,] 0.09140746 0.1916180
```

```
## [2,] 0.09325416 0.1981803
##
## [[9]]
##           [,1]      [,2]
## [1,] 0.07257985 0.1537165
## [2,] 0.07432195 0.1587617
##
## [[10]]
##           [,1]      [,2]
## [1,] 0.05779083 0.1231842
## [2,] 0.05931687 0.1271181
```

- c) Draw a corresponding innovation sequence at for 300 periods from a (multivariate) Gaussian distribution and simulate the given VAR(1) process without any further built-in functions

Hint: 'mvrnorm' and 'for' are still allowed

Solution:

1. Set T (in code N)
2. Draw $\{a_1, \dots, a_T\} \sim [\mu, \Sigma_a]$
3. Set $z_0 = \mathbb{E}(z_t)$
4. $z_1 = \phi_1 z_0 + a_1$
5. Repeat Sep 4 ($T - 1$) times
6. (Discard first few observations to minimise effects of z_0 on the results)

\Rightarrow *Note* that it is generally advised to discard the first few observations to eliminate the influence of the arbitrary starting point! In our case we skipped this step to keep things short and clear.

```
# i) Set sample size and further coefficients
N <- 300
# ii) The Basis: Innovations
set.seed(2^9-1) # for replication: the 'random' numbers drawn
# up from this point will always be the same (given you use
# the same command to draw them....)
```

```

a <- mvrnorm(n = N, mu = c(0,0), Sigma = Sigma_a)
# iii) Writing a function to generate 'z' realisations from 'a' using Phi
varlgen <- function(coef.mat, z.lag, innovation){
  z.current <- coef.mat %*% z.lag + innovation
  #  $z_{-}(t) = \phi_i * z_{-}(t-1) + a_{-}(t)$ 
  return(z.current)
}
# iv) Prepare variable for 'z'
z <- ts(data = matrix(data = NA, nrow = (N), ncol = k)
        ,start = 1, names = c("z_1", "z_2"))
# v) Set starting values for z (at mean + innovation)
z[1,] <- c(0,0) + a[1,]
# vi) Generate z repeatedly and store it
for (i in 2:(N)){
  z[i,] <- varlgen(coef.mat = Phi, z.lag = z[(i-1),], innovation = a[i,])
}

```

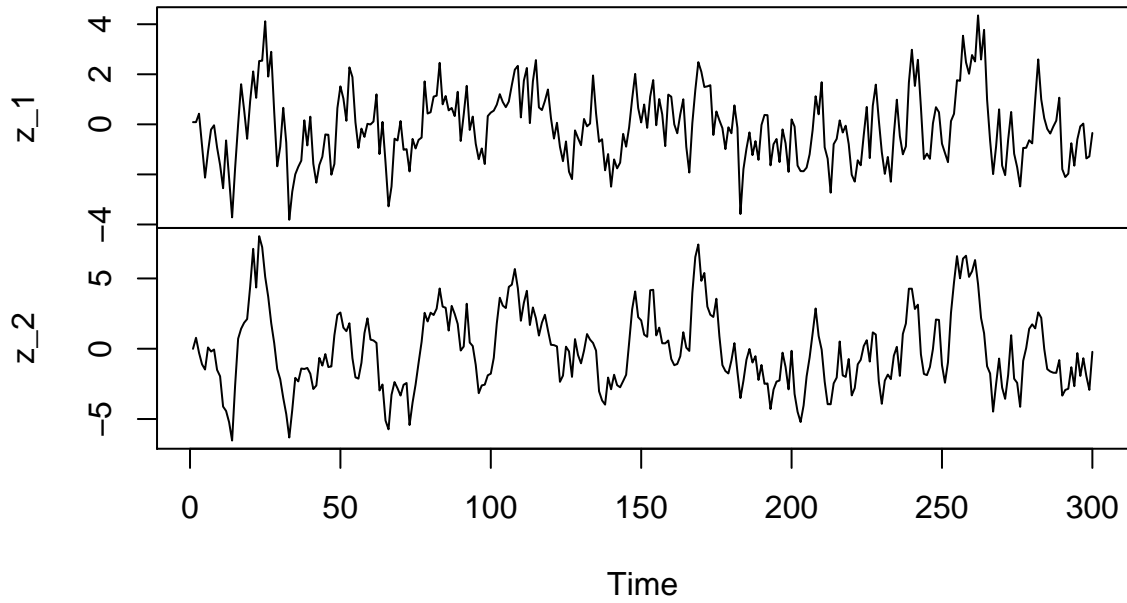
d) Plot the multivariate time series you have just created. Does it look stationary?

```

# The ordinary 'plot' command works because 'z' is already
# a 'ts'-class object (among other classes) and NOT a data frame!
plot(z)

```

z



At least it looks stable hence we cannot rule out stationarity.

- e) Estimate the sample cross-covariance and cross-correlation matrices. Compare these with the population moment matrices from task a)

$$\widehat{\Gamma}_0 = \tilde{z}'_T \tilde{z}_{T-1} \cdot (T-1)^{-1}$$

$$\tilde{z}'_T = z'_T - \hat{\mu}_z$$

Z is a $T \times 2$ matrix

z_t is a 2×1 vector

$$z_t := \begin{pmatrix} x_t \\ y_t \end{pmatrix} \leftarrow \text{variables}$$

$$Z_t := \begin{pmatrix} X_t & Y_t \\ X_{t-1} & Y_{t-1} \\ X_{t-2} & Y_{t-2} \\ \vdots & \vdots \\ X_{t-t} & Y_{t-t} \end{pmatrix} \leftarrow \text{sample data}$$

$$\widehat{\text{Cov}}(x_t, y_{t-1})$$