Chapter 5
Vector Models for Multivariate Problems

5.1 Introduction and VAR models

While univariate models are useful foundation for time series modeling, economists almost immediately need multivariate or vector models. Thus vector versions of ARIMA models may be called VecARIMA(p, d, q) upon simplifying by removing the seasonal part. It is clear by analogy with univariate situation that inclusion of both AR and MA components in a model leads to parsimony, as stated in Remark 2.4.1. This insight from chapter 2 carries forward to vector models with VecARMA(p, p-1) models representing vector dynamics of order p in this chapter.

Economists quickly discover that two variables are not enough. For example, the Fisher equation in macroeconomics states that after tax nominal interest rate adjusted for expected inflation should equal the real interest rate. See (3.2.11) to (3.2.13). It may appear that nominal interest rate and inflation rate are the only two variables, but since the marginal tax rate on interest earnings does not remain constant over time, a proper test of the Fisher equation needs to allow for the third variable $\tau_t$ for marginal tax rate and replacing nominal interest rate variable $i_t$ in all three equations by $i_t - \tau_t$. The Fisher theory suggests that economic agents fully subtract the tax rate, but whether they do so in practice is an empirical question and there may be a need for a parameter $\gamma$ leading to the use of $(i_t - \gamma \tau_t)$ instead to $i_t$.

A non-theoretical flexible and efficient method to jointly study all three variables and their lags in a parsimonious model is to write a vector autoregressive moving average model VecARMA(p, q). Vinod (1987) argued that structurally modified VecARMA(p, p-1) model is ultimately the right way to go. Unfortunately, whenever moving average terms are admitted, the estimation involves complicated nonlinear conditional maximum likelihood methods and presents several practical difficulties. There are problems of multiple solutions, dependence on starting values, non-uniqueness, unexpected signs, etc. Over time, as software tools improve, these problems are becoming more manageable. For example, R software offers the package “dse” to estimate VecARMA models, but the implementation is not at all easy. For the time being VecAR(p) or VAR(p) models remain as suitable stopgap tools. Moreover, we shall see that the VAR viewpoint offers convenient statistical testing of multiple cointegrating relations and plotting of impulse response functions.

Sims (1980) challenged the Cowles Commission modeling of structural simultaneous econometric models and advocated atheoretical (devoid of theory) modeling for macroeconomics. He argued that economic theory does not really provide enough details for empirical specification of macroeconomic relations. His vector autoregression (VAR) analysis provides modeling based on purely statistical relations among the following
variables money (M), real GNP (RGNP), unemployment rate (U), wage rate (W), price level (P) and import price index (IP). Simplification based on prior non-sample information leads to structural VAR or SVAR models.

5.1.1 Some R packages for vector modeling
The contributed R package “sem” written by John Fox offers functions for fitting general structural equation models by maximum likelihood and instrumental variables methods including two-stage least squares (TSLS).

The contributed R package “pls” offers functions for multivariate regression. Principal Component Regression (PCR) and two types of Partial Least Square Regression (PLS), simple-PLS and kernel-PLS, are implemented by Ron Wehrens.

fMultivar package offers Technical Analysis, Benchmark Analysis, Rolling Analysis, Regression Modelling and tests, Simultaneous Equations Modeling and some useful tools for matrix algebra and multivariate densities.

The ‘vars’ package comes with a useful vignette describing its content in full detail. It is comprehensive and easy to use. It provides tools for estimation, imposing restrictions, diagnostic testing, forecasting, impulse response analysis (IRA) and forecast error variance decomposition (FEVD) similar to analysis of variance for VAR and structural VAR models.

5.1.2 VAR models

A VAR model consists of a vector of K endogenous variables, \( y_t = (y_{1t}, \ldots, y_{kt}, \ldots, y_{Kt}) \) for \( k = 1, \ldots, K \). One defines a VAR(p)-process as

\[
y_t = A_1 y_{t-1} + \ldots + A_p y_{t-p} + u_t.
\]

(5.1.1)

with \( p \) vectors of lagged values and \( p \) matrices \( A_i \) of dimensions \( K \times K \) for \( i = 1, \ldots, p \), and where \( u_t \sim \text{WN}(0, \Sigma) \), a \( K \)-dimensional white noise process with \( \Sigma \) denoting a time invariant positive definite covariance matrix. If \( K=2 \) we have the bivariate case of chapter 3. The VAR(2) model in the bivariate case is:

\[
y_{1t} = A_{1,11} y_{1,t-1} + A_{1,12} y_{2,t-1} + A_{2,11} y_{1,t-2} + A_{2,12} y_{2,t-2} + u_{1t},
\]

(5.1.2)

\[
y_{2t} = A_{1,21} y_{1,t-1} + A_{1,22} y_{2,t-1} + A_{2,21} y_{1,t-2} + A_{2,22} y_{2,t-2} + u_{2t}.
\]

where \( A_1 \) and \( A_2 \) are \( 2 \times 2 \) matrices with elements along first row with additional subscripts (11, 12). The additional subscripts along second row are (21, 22). The \( 2 \times 1 \) vector \( y_t \) is bivariate with elements \( y_{1t} \) and \( y_{2t} \), and similarly for the \( u_t \) vector. The lagged values can be replaced by the \( L \) operator and it is possible to write a matrix polynomial in the lag operator \((1 - A_1 L - A_2 L^2)\) where \( I \) is identity matrix of order 2. Unit root means that the determinant of the matrix polynomial has root \( L=1 \). We have a stationary stable VAR process if the determinant is nonzero with roots \(|\lambda_i|<1\), inside the unit circle. VAR(p) model is a direct generalization of (5.1.2) with \( p \) instead of 2 vector elements with no additional conceptual difficulty.
Consider a macroeconomic model for US consisting of series used by Sims (1980) in his Econometrica article, which started the popularity of VAR models. He used variables named as M (=M1 money supply), Y (=Nominal GNP), U (=unemployment rate), W (=average compensation), P (=GNP price deflator as a measure of inflation) and PM (=import prices) over 1949 to 1975 for US. The following subsection is devoted to data collection tips in R

5.1.3 Data Collection Tips Using R

The modeling exercise starts with data collection into comparable time series. This task has become somewhat easier in recent years due to Internet availability of data. However considerable care is needed to prepare the data for analysis. This section helps the reader in these tasks.

Variable names at Economagic website are useful to know for direct download from there. The names are not at all intuitive and as of this writing there is no place where they are conveniently described.

TABLE of Economagic names and their meaning.
beana/t102101 =Real Gross Domestic Product
fedstl/trsp500 SP 500 Total Return
fedstl/gnp =Gross National Product in Current Dollars
var/cpiu-long =Consumer Price Index - All Urban Consumers
feddal/ru =Unemployment Rate
fedstl/indpro =Total Industrial Production Index
fedstl/exjpus+2 =FX Rate: Japanese Yen to one US Dollar
fedstl/fedfunds+2 =Federal Funds Rate
fedstl/mdiscrt+2 =Discount Rate
fedbog/tcm30y+2 =30-Year Treasury Constant Maturity Rate
fedstl/mprime+2 =Bank Prime Loan Rate
fedstl/tb3ms+2 =3-Month Treasury Bills - Secondary Market

Since they are for free download, one cannot ask for help from anyone and there are no guarantees. After one downloads them, it is a good idea to keep them in storage in case the website is shut down for some reason. They need the R package ‘fCalendar’ for proper download functions. The following R snippet describes the downloads of various data needed here. The ‘head’ and ‘tail’ functions in R are helpful for seeing the data at the beginning and end of the series to make sure what the data looks like and summary function helps locate missing values and strange things.

```R
#R5.1.1
library(fCalendar)
#Get quarterly unemployment data from Economagic
# # # # # # # #
USU = economagicImport(query = "feddal/ru",
  frequency = "quarterly", coiname = "ru")
  # Print Data Slot if Internet Download was Successful:
U=USU@data #the at symbol is counterintuitive peculiarity of economagic head(U);tail(U); summary(U)
```
The Data do not seem to be quarterly at all but are monthly despite the part of the call stating that (frequency = “quarterly”). Economagic seems to ignore it. We need a way (algorithm or function) to convert monthly data to quarterly allowing missing data as NAs. For some series like the unemployment rate, the conversion should average the monthly rates over three months to obtain quarterly rate. For some series like number of housing starts the quarterly data is a cumulative sum of three month data. We need a function which allows for this option. The following illustrates the use of functions in R. It is written in a way that cumulation is implicit and averaging is optional set by setting cum=F when calling the function named ‘m2qNAappend’ where ‘m2q’ means convert monthly to quarterly. If the data length is not a multiple of 12 for monthly data, the function tries to append right number of NAs to make it so. This might not work well and the user is warned that R has the feature (nasty habit?) of recycling the available numbers to fill the data gaps when it constructs matrices by the command ‘matrix’ and the user supplies insufficient elements. This function avoids loops, which are a ‘No-no’ for professional R programmers since they slow things down. It uses the apply function in R to avoid loops. The apply function applies some function such as mean or sum to rows or columns if the second argument is 1 or 2. In the following it is 1. The print statements are commented out. The user should remove the comment symbol “#” from the print statements, work on a small example and make sure that it is doing exactly what is intended. A possible example is given at the bottom of the function with # symbols. We use these # symbols so that the user can copy and paste all lines into R.

#R5.1.2 R function to convert monthly data to quarterly
#works by appending NAs to make data length a multiple of 12
m2qNAappend=function(x, cum=T, start.month=1, end.month=12){
  #If you want to average over the months set cum=F
  y2=x
  if (start.month !=1) y2=c(rep(NA,(start.month-1)),x)
  #print(y2)
  if (end.month !=12) y2=c(y2, rep(NA,(12-end.month))))
  #print(y2)
  n=length(y2)
  #Version attaches NAs at the end till n is multiple of 3
  n1=n%%3; y=n1+1
  if (n>0){y=c(y2, rep(NA,n2));n1=n1+1}
  y=matrix(y,nrow=3, ncol=n1);
  #print(y)
  if (cum) quar=apply(y,2,sum, na.rm=T)
  if (cum=F) quar=apply(y,2,mean, na.rm=T)
  list(quar=quar)
}
#example #x=1:18; x[3]=NA; m2qNAappend(x,start.month=2)

Our next task is to apply the above function to the unemployment data. The way the function is written, it will create a list with name quar for quarterly data. Note the use of unem$quar to extract the quarterly series and the ts function to define time series properties.

#R5.1.3 Load R5.1.1 and R5.1.2 into memory before using this
unem=m2qNAappend(as.numeric(U$ru),cum=F)
unem=ts(unem$quar, frequency=4, start=c(1969,3))
plot(unem, main="US Quartely Unemployment Rate")

So far we have U (=unemployment rate) from Sims’ (1980) list. We still need M (=M1 money supply), Y (=Nominal GNP), W (=average compensation), P (=GNP price deflator as a measure of inflation) and PM (=import prices). The following code collects those series.

#R5.1.4
## Get US GNP   # # # # # # # #
USGNP = economagicImport(query = "fedstl/gnp", 
                         frequency = "quarterly", colname = "USGNP")
GNP=USGNP@data
head(GNP);tail(GNP); summary(GNP)
gnp=ts(as.numeric(GNP$USGNP), frequency=4, start=c(1947,3))
plot(gnp, main="US Nominal GNP")

## Get US Money supply M1  # # # # # # # #
USM = economagicImport(query = "fedstl/m1sl", 
                        frequency = "quarterly", colname = "USM")
M=USM@data
head(M);tail(M); summary(M)
money=m2qNAappend(as.numeric(M$USM),cum=F)
money=ts(money$qua, frequency=4, start=c(1959,1))
plot(money, main="US Money Supply M1")

## Get US GDP implicit price deflator  # # # # # # # #
USP = economagicImport(query = "fedstl/gdpdef", 
                        frequency = "quarterly", colname = "USP")
P=USP@data
head(P);tail(P); summary(P)
pgnp=ts(as.numeric(P$USP), frequency=4, start=c(1947,1))
plot(pgnp, main="US GNP Implicit Price Deflator")

Implicit price deflator is a measure of inflation. However we have to decide which is the base year. If we give the R command P[210:230,] we get the following output:

<table>
<thead>
<tr>
<th>DATE</th>
<th>USP</th>
</tr>
</thead>
<tbody>
<tr>
<td>210 1999-12-01</td>
<td>98.432</td>
</tr>
<tr>
<td>211 2000-03-01</td>
<td>99.317</td>
</tr>
<tr>
<td>212 2000-06-01</td>
<td>99.745</td>
</tr>
<tr>
<td>213 2000-09-01</td>
<td>100.259</td>
</tr>
<tr>
<td>214 2000-12-01</td>
<td>100.666</td>
</tr>
</tbody>
</table>

This output suggests that the base value 100 is between quarter 3 and 4 in year 2000. When we use inflation adjustment for GNP and wages we can divide by the price index. This changes the units of measurement of GNP and wages since we are dividing by 100 times the proper number. A simple way to handle this is to multiply the ratio by 100 to keep original units. The command is ‘realgnp=(100*gnp)/pgnp’. The data on import prices and wages required much work to get in the proper format. We simply provide the series after the manipulations.

#R5.1.5
# READ Import Price data into memory # # # # # # # #
# Source ftp://ftp.bls.gov/pub/time.series/ei/ei.data.4.BEA
ImpPrice = c(80.0, 79.9, 77.7, 77.8, 77.6, 78.0, 78.7, 78.9, 77.7, 77.0, 75.5, +75.8, 76.0, 77.8, 75.0, 75.1, 76.9, 77.7, 80.8, 83.6, 84.3, 85.5, 86.5, +88.8, 87.6, 89.3, +91.30000, 92.03333, 90.30000, 91.20000, 93.43333, +98.80000, 95.66667, 93.86667, 93.06667, 94.30000, 93.96667, 94.13333, +95.63333, 96.66667, 94.40000, 95.23333, 94.46667, 94.26667, 93.80000, +95.50000, 97.33333, 98.26667, 99.33333, 101.36667, 100.90000, 100.66667, +101.26667, 101.66667, 101.03333, 102.36667, 101.13333, 98.80000, 98.43333, +97.96667, 95.00000, 93.56667, 92.16667, 91.70000, 91.33333, 92.80000, +94.80000, 96.70000, 99.13333, 99.16667, 100.73333, 100.96667, 99.56667, +97.80000, 96.00000, 92.46667, 92.00000, 94.26667, 94.93333, 95.10000, +98.16667, 95.83333, 96.53333, 96.86667, 99.53333, 101.33333, 103.26667, +105.10000, 105.96667, 108.63333, 112.33333, 113.03333, 113.06667, 116.53333, +117.73333, 114.06667)  
ImpPr = ts(ImpPrice, frequency = 4, start = c(1982, 3))  
plot(ImpPr, main="Import Prices")

# Read nominal wage data into memory # # # # # # # #
# http://www.bls.gov/bls/glossary.htm#earnings
# wage data converted into quarterly data from sources  
http://data.bls.gov/PDQ/servlet/SurveyOutputServlet  
wage = c(102690.7, 102915.3, 103249.0, 103417.7, 103584.3, 103838.0, 104127.3, 104427.7, +104733.3, 105020.3, 105248.3, 104982.3, 104692.3, 104506.7, 104542.7, 104746.7, +104863.3, 105006.7, 105342.7, 105703.0, 106672.0, 106904.7, 107139.7, 107503.3, +107876.7, 108177.0, 108443.3, 108786.0, 109130.3, 109533.7, 109883.7, 110186.0, +110483.3, 110787.7, 111113.3, 111431.0, 111720.3, 112045.3, 112430.7, 112865.7, +113236.3, 113532.0, 113846.3, 114283.3, 114714.3, 115139.0, 115550.7, 115918.0, +116707.7, 117036.7, 117411.0, 117824.3, 118254.3, 118636.0, 119000.7, 119189.7, +119378.7, 119819.3, 120368.0, 121045.7, 121640.0, 122166.7, 122669.7, 123188.7, +123708.0, 124203.0, 124739.3, 125289.0, 125814.0, 126324.7, 126745.0, 127169.3, +127511.3, 127868.7, 128233.7, 128617.0, 129043.7, 129527.0, 130165.7, 130757.3, +131267.0, 131712.3, 132250.0, 132880.0, 133476.0, 134020.3, 134595.0, 135246.7, +135949.7, 136676.7, 137456.0, 138260.3, 139033.7, 139827.3, 140602.7, 141401.7, +143005.3, 143758.7, 144522.7, 145215.0, 145964.3, 146719.7, 147478.3, 148226.0,
+148986.7, 149746.7, 150498.0, 151253.0, 151987.3, 152707.7, 153579.0, 154336.3, 155075.0, 155773.7, 156526.7, 157222.0, 157910.7, 158652.3, 159429.7, 160140.3, 160828.7, 161525.3, 162265.0, 163024.0, 163756.3, 164447.3, 165199.7, 166054.7, 166762.3, 167415.7, 168110.7, 168693.7, 169279.0, 169837.3, 170412.7, 170990.3, 171497.0, 172020.0, 172521.7, 173046.0, 173505.0, 173957.3, 174449.3, 174950.3, 175678.7, 176125.3, 176595.3, 177132.3, 177522.3, 177946.3, 178413.3, 178940.7, 179825.3, 180320.7, 180835.7, 181365.3, 182001.3, 182526.7, 183016.0, 183467.0, 183967.3, 184389.3, 184840.3, 185253.3, 185772.7, 186178.0, 186602.3, 187017.7, 188519.7, 188916.3, 189352.7, 189866.3, 190271.7, 190655.7, 191121.3, 191650.7, 192074.7, 192506.7, 193024.3, 193615.7, 194106.0, 194555.3, 195068.0, 195621.0, 196085.3, 196522.0, 197050.0, 197600.7, 197882.0, 198295.7, 198807.0, 199351.7, 199776.0, 200279.3, 200849.7, 201457.3, 202395.7, 202835.3, 203366.7, 203935.3, 204395.0, 204905.0, 205482.7, 206097.7, 206876.0, 207431.7, 208043.7, 208660.3, 211586.0, 212242.0, 212918.7, 213560.3, 214101.0, 214735.7, 215421.7, 216111.7, 216664.0, 217203.7, 217867.7, 218543.0, 220109.3, 220774.0, 221512.7, 222275.7, 222356.0, 222973.3, 223680.0, 224418.0, 225038.0, 225674.0, 226422.3, 227196.0, 227763.7, 228432.7, 229166.3, 229896.0

# quarterly US wages from 1948 Q1 to 2006 Q4  59 years 236 observations
wage=ts(wage, frequency=4, start=c(1948,1))
plot(wage, main="US Nominal Wages")

# Construct inflation adjusted real wage and real GNP data
# pgnp the inflation measure is an index with base 100
realwage=(100*wage)/pgnp
realgnp=(100*gnp)/pgnp

library(vars)
layout(matrix(1:4, nrow = 2, ncol = 2))
plot(unem, main="US Quartely Unemployment Rate")
plot(gnp, main="US Nominal GNP")
plot(realgnp, main="US Real GNP")
plot(money, main="US Money Supply M1")
#R5.1.6
layout(matrix(1:4, nrow = 2, ncol = 2))
plot(pgnp, main="US GNP Implicit Price Deflator")
plot(ImpPr, main="Import Prices")
plot(wage, main="US Nominal Wages")
plot(realwage, main="US Real Wages")
R5.1.7 Sims' order is M Y U W P PM we do the same in our notation
usdata=ts.intersect(money, realgnp, unem, realwage, pgnp, ImpPr)
#ts.intersect is powerful command to get right year and quarter
#but it adds two columns for year and quarter and VAR
#command is not set up to recognize such time series. It expects
#an input of a matrix
nr=nrow(usdata)
cnam=colnames(usdata)

#usdata=ts.intersect(unem, log(realgnp), log(money), log(pgnp),
#log(ImpPr), log(realwage))

usdata=as.matrix(usdata)# need for this was recognized
# after a lot of trial & error. Otherwise the VAR command does not work.
colnames(usdata)=cnam  # even after making it a matrix get same names

#nrow(usdata)
#rownames(usdata)=as.character(1:96)
5.1.4 VAR Estimation of Sims’ Model

The R function for estimating a vector autoregression model VAR is VAR in the package vars. 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). 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.

```r
VARselect(usdata, lag.max = 5, type = "const")
#$selection  #AIC(n) HQ(n) SC(n) FPE(n)
# we select lag=2
#AIC(n)  HQ(n)  SC(n) FPE(n)
#    3      2      1      3
var.2c <- VAR(usdata, p = 2, type = "const")
summary(var.2c)# notation 2c for p=2 and const
plot(var.2c)
roots(var.2c)# stable model has all roots <1
```

Before we report the detailed results we want to make sure that they are suitable representations of the US macro economy. The moduli of the eigenvalues of the companion matrix need to be less than one. In our case the first moduli (= 1.011) exceeds 1 suggesting that we have an unstable nonstationary or evolving system. This also means we may consider changing Sims’ specification. Potential choices include p=3, logs of variables, shorter list of variables, inclusion of federal funds rate as a variable and so on.
5.1.5 Granger Causality Analysis in VAR models

Granger (1969) proposed certain tools for detection of causalities between two or more economic variables. Since they cannot reveal true causality, any more than any statistical method will reveal true causality, the literature refers to them as Granger-Causality tests. The reason is simply that any statistical test based on lags and leads of effects is subject to the famous fallacy called “post hoc ergo propter hoc”. The ancients were aware that a rooster shouting the sunrise everytime before sunrise is not the cause of sunrise.
Roughly speaking, we say that the variables \( x \) does granger-cause variable \( y \) if variable \( x \) helps to predict variable \( y \). The idea can be formulated in terms of mean square error (MSE) of forecasting \( h \) step ahead based on information set \( I_{nf,t} \) at time \( t \). For example, we say that \( x \) does Granger-cause \( y \) if the MSEs satisfy the inequality:

\[
MSE_h(I_{nf,t,\text{with } x}) < MSE_h(I_{nf,t,\text{without } x})
\]

(5.1.3)

where we have spelled out the condition on the information set whether or not it excludes information in the \( x \) variable. Let us rewrite (5.1.2) by changing \( y_1 \) to \( y \) and \( y_2 \) to \( x \) as:

\[
\begin{align*}
    y_t &= A_{1,11} y_{t-1} + A_{1,12} x_{t-1} + A_{2,11} y_{t-2} + A_{2,12} x_{t-2} + u_{1t}, \\
    x_t &= A_{1,21} y_{t-1} + A_{1,22} x_{t-1} + A_{2,21} y_{t-2} + A_{2,22} x_{t-2} + u_{2t}.
\end{align*}
\]

(5.1.4)

One can understand the logic behind Granger-causality by considering whether \( x \) is useful in forecasting \( y \) without using information in variable \( x \). Thus \( x \) does NOT cause \( y \) means omitting the terms \( A_{1,12} x_{t-1} \) and \( A_{2,12} x_{t-2} \) from the first part of (5.1.4), that is assuming that the regression coefficients \( A_{1,12} = 0 \) and \( A_{2,12} = 0 \). Thus in a VAR(2) model the second variable \( x \) does not Granger-cause the first variable \( y \) if and only if the coefficients \( A_{i,12} \) for \( i=1,2 \) are statistically insignificant. This is readily implemented by focusing on the regression having the ‘effect’ variable as the dependent variable and considering the F test for the joint null hypothesis that the coefficients of ‘cause’ variable are all zero.

**Definition:** In general, in a VAR(p) model the \( j \)-th ordered variable does NOT Granger-cause the \( k \)-th ordered variable if and only if the joint null hypothesis that \( A_{i,kj}=0 \) for all \( i=1,\ldots,p \) is not rejected by the data.

The VecMA(\( \infty \)) representation of AR models is used to further generalize this definition.

The vars package has the function causality() to implement a Wald-type causality test. It systematically tests for nonzero correlation between the error processes of the cause and effect variables. The R package called dynlm is for dynamic linear models. The package mAr implements Stepwise least-squares estimation of a multivariate AR(p) model based on the algorithm of Neumaier and Schneider (2001). The Bayesian package MSBVAR does Baysian vector AR. The reader should be aware of potential misuses of Granger causality. If only a few variables are included in the VAR model, the omitted variables may be confounding the causal relations estimated by Granger-causality conclusions. Confounding is known as Simpson’s paradox in statistical literature. In the present context confounding might well arise from ignoring seasonal effects. One may also be subject to measurement errors.

**Instantaneous Causality:** If \( x_t \) does Granger cause \( y_t \) immediately, would it show in regression coefficients among a version of (5.1.4) with contemporaneous \( x \) and \( y \) also appearing on the right side? If true causation has no time lags, then it is impossible to determine which is the cause and which is the effect by looking at coefficients. That is, we have an identification problem. However, the covariance matrix among errors \( \Sigma_u \) in (5.1.4) contains useful information about absence of instantaneous causation, namely that
its off-diagonal elements are zero. See Lutkepohl (2005, Sec. 2.3 and 3.6) and explanations of the R package ‘vars’ for explicit formulas.

The ‘vars’ package illustrates Granger causality analysis with Canadian data. The following R snippet is implemented next. We are unable to illustrate with Sims’ data by using the command ‘causality(var.2c, cause = “realgnp”)’ since Sims’ model VAR system is singular!

```r
#R5.1.9
library(vars); data(Canada)
#first fit the VAR model to the data choosing lag order=2
#const will include the deterministic regressor column of ones
var.2c <- VAR(Canada, p = 2, type = "const")
#output is an R object named (.2c) for 2 lags and c for const
causality(var.2c, cause = "e")
```

Output of R5.1.9 snippet.
The dataset ‘Canada’ is supplied by the package. It is quarterly data from 1980 to 2000, where ‘prod’ measures labor productivity, ‘e’ represents employment, ‘U’ is the unemployment rate, and ‘rw’ represents the real wage.
1) Null: employment ‘e’ does not Granger-cause remaining 3 variables: prod, rw, and U.

F-Test = 6.2768, df1 = 6, df2 = 292, p-value = 3.206e-06

2) Null: employment ‘e’ does not instantaneously cause the remaining 3 variables: prod rw U. The Wald test statistic is asymptotically Chi-square.

\[ \text{Chi}^2 = 26.0685, \text{df} = 3, \text{p-value} = 9.228e-06 \]

The very low p-values in both types of causality tests (Granger and instantaneous) we reject non-causality, supporting possible causation. We have determined a causal relation, because there are macroeconomic reasons to believe in a causal relation is at all plausible. That is, we are not subject to the ‘post hoc’ fallacy here.

5.1.6 Forecasting Out-of-Sample in VAR models

The out-of-sample forecasting activity is obviously important in time series analysis and VAR models are no exception. One reserves a certain number (say m) of observations for out-of-sample forecasting and estimates the VAR model with only \( T - m \) observations. If Brenard Pfaff’s ‘vars’ package is used, its ‘VAR’ function will create an output object (named var.2c above). Next, another function called ‘predict’ provides h-step ahead forecast with a flexible choice of \( h=1, 2, \ldots, n \), where \( n \) is chosen under the name ‘n.ahead.’ It simply plugs in the past values recursively to get subsequent (future) values. It also provides a confidence interval around the forecasts. In the following snippet we use all observations (\( m=0 \)), go 10 steps ahead and construct a 95% forecast interval.
#R5.1.10
var.f10 <- predict(var.2c, n.ahead = 10, ci = 0.95)
names(var.f10)
layout(matrix(1:3, nrow = 3, ncol = 1))
plot(var.f10)
#fanchart(var.f10) This is an alternative command
5.1.7 Impulse response analysis in VAR models

Engineers have long studied the response of one variable in the form of a short impulse to another variable, where the engineer often explicitly administers the impulse to a tangible machine and measures the response. For example, if a driver briefly presses the gas pedal to overtake a truck, the automobile will temporarily go faster, but after a time lag return to the original speed. Engineers also study ‘step response,’ where the response is measured with the input kept at a higher level (driver stepping up the gas) over time. Step response functions are rarely, if ever, mentioned in econometrics.

In economic applications we work with less tangible phenomena and imagine our impulses as simply ‘innovations’ or changes in a variable occurring in historical data. Of course, when the central bank (the ‘Fed’) changes the federal funds rate by 50 basis points (say,.) it is somewhat akin to an artificially administered impulse. A study of the
effect of all such impulses (artificially administered or passively observed in historical data) on the economy is profoundly important and worthy of study. It goes well beyond analysis of Granger-causality discussed above.

The impulse response function (IRF) is a tool developed by engineers to study the response of a dynamic system to a unit change in one variable. The effect on the system consists of immediate effect as well as delayed effect on all variables. The IRF is usually a plot with time lags on the horizontal axis and the relevant effect (simple, cumulative or orthogonal, as explained below) on the vertical axis. For example, Sims (1980) used IRFs to investigate the dynamic interactions of various variables to shocks or innovations to other variables. The IRF was mentioned in the univariate context in Section 2.2 above as Green’s function as measuring \( \frac{\partial y_t}{\partial a_{t-j}} \), the effect of a unit perturbation (impulse) in innovation at lag j denoted there by \( (a_{t-j}) \) on \( y_t \).

Many time series books (e.g., Lutkepohl, 2005, Sec. 2.1) rewrite a stationary stable VAR(p) system of (5.1.1) having k variables into an equivalent VAR(1) model with kp variables. This involves stacking of the dependent variable \( y_t \) into kp\( \times \)1 vector \( Y_t \) with k elements for each of the p lags: \( Y_t = [y_t, y_{t-1}, .. y_{t-p+1}]' \). Also, we stack adequate number of zero vectors below the error \( u_t \) of (5.1.1) to create a conformable kp\( \times \)1 vector of errors \( \varepsilon_t \). Now VAR(p) is equivalent to the following VAR(1):

\[
Y_t = A Y_{t-1} + \varepsilon_t, \tag{5.1.5}
\]

with a large kp\( \times \) kp matrix \( A \) having the first row containing all the coefficient matrices \( A_1 \) to \( A_p \) in (5.1.1). For example, if p=3 we have the following special case of (5.1.5):

\[
\begin{pmatrix}
  y_t \\
  y_{t-1} \\
  y_{t-2}
\end{pmatrix}
= \begin{bmatrix}
  A_1 & A_2 & A_3 \\
  I & 0 & 0 \\
  0 & I & 0
\end{bmatrix}
\begin{pmatrix}
  y_{t-1} \\
  y_{t-2} \\
  y_{t-3}
\end{pmatrix}
+ \begin{pmatrix}
  u_t \\
  0 \\
  0
\end{pmatrix}. \tag{5.1.6}
\]

In general, \( A \) has a large k(p-1)\( \times \) k(p-1) identity matrix along the p-1 columns of \( A \) and the last column has all zeros. This rewriting is useful in explaining how moving average representation yields impulse responses and how powers of the matrix \( A \) arise, Lutkepohl (2005, p.52). Our brief intuitive description hopes to make the omitted tedious algebra plausible. Recall that Chapter 2 studied the stochastic difference equations and noted in the discussion following eq. (2.2.5) that the solution of AR(1) at time t is \( y_t=\phi y_0 \), where \( \phi \) is the root for the first order and represents its MA(\( \infty \)) representation. Eq. (2.2.11) provides the IRF. Univariate AR(p) system will have p terms associated with p roots of its polynomial. A device analogous to (5.1.5) and (5.1.6) can convert AR(p) to multivariate AR(1) system. In this chapter with VAR(p) models, algebra is more complicated but essentially analogous. Here too, if the system starts at time 0 with stacked vectors of starting values \( Y_0 \), as time passes they become A\( Y_0 \) at time i, where the powers of the large \( A \) matrix of (5.1.5) absorb the p terms in the MA representation.

In general, the impulse response function (IRF) for VAR models is obtained from the MA(\( \infty \)) representations for VAR(p)-process. For non-stationary or cointegrated systems, if the MA representation is unavailable, alternatives are developed in the literature. For
example, vector error correction models (VECM or Johansen’s canonical correlations) are used instead of VAR. The study of IRFs is feasible, even if Granger causality could not be implemented for our version of extended Sims’ data. The ‘vars’ package computes them in the form of the expected response of variable $y_{i,t+s}$ to a unit change in variable $y_j$. These effects can be retained as they are (simple) or cumulated through time to obtain the cumulated impact of a unit change in variable $j$ to the variable $i$ at time $s$.

**Orthogonalized impulse responses:** These are useful if the reader suspects that the underlying shocks do not occur in isolation, but jointly due to contemporaneous correlations between errors for different variables. However for policy analysis or other purposes the focus may be on exclusive impact of one variable after removing other effects. Engineers have solved this problem long ago by using a matrix algebra device. Accordingly the off-diagonal elements of such correlation matrix are made zero by using a Choleski decomposition of the error variance-covariance matrix $\Sigma_u = PP'$ where $P$ is a lower triangular matrix and $P'$ denotes its transpose. If the equation errors are $u_t$, they are changed to $\epsilon_t$ by the transformation $\epsilon_t = P^{-1}u_t$ by using the inverse of the Cholesky decomposed $P$ matrix. The final IRF is obtained by plugging these revised innovations into the suitable MA representation. The ‘vars’ package readily offers this feature.

```r
#5.1.11
irf2c=irf(var.2c)# run the impulse response function
length(irf2c$irf$mo)# select the table of money innovations
#cnam ="money" "realgnp" "unem" "realwage" "pgnp" "ImpPr"
nstart=seq(1,66,11)
nend=11*1:6

layout(matrix(1:3, nrow = 3, ncol = 1))
for ( i in 1:3){
plot.ts(irf2c$irf$mo[nstart[i]:nend[i]], main="Impulse response of money innovations", ylab=cnam[i])}

for ( i in 4:6){
plot.ts(irf2c$irf$mo[nstart[i]:nend[i]], main="Impulse response of money innovations", ylab=cnam[i])}

#Now plot impulse response of import price innovations
layout(matrix(1:3, nrow = 3, ncol = 1))
for ( i in 1:3){
plot.ts(irf2c$irf$ImpPr[nstart[i]:nend[i]], main="Impulse response of import price innovations", ylab=cnam[i])}

for ( i in 4:6){
plot.ts(irf2c$irf$ImpPr[nstart[i]:nend[i]], main="Impulse response of import price innovations", ylab=cnam[i])}
```
Impulse response of money innovations

Impulse response of money innovations

Impulse response of money innovations
Impulse response of money innovations

Impulse response of money innovations

Impulse response of money innovations
Note that we have reported the impulse responses of the US macro economy to innovations in two variables: ‘money’ and ‘ImPr’, which are potentially subject to policy changes. After all, the Federal Reserve can influence the money supply, and prices of imports can be influenced by policies including tariffs. The responses depicted over 12 quarters or three years are worthy of study as interesting insights from a non-theoretical model based purely on publicly available data. The R package allows depiction of responses of any variable to any other variable in the list of variables chosen by the researcher. By simultaneously considering all six variables at the same time in a VAR system, we are obviously measuring the lagged responses more accurately than bivariate models. As with Granger causality, impulse response analysis is subject to Simpson’s paradox problem arising from the ‘post hoc’ fallacy, confounding of omitted variables (or seasonal effects) and measurement errors.
5.2 Multivariate Regressions: Canonical Correlations

Hotelling (1936) was the first to discuss relations between two sets of variables by using matrix algebra results related to eigenvalues and eigenvectors. Certain eigenvalues are transformed and interpreted as correlation coefficients and corresponding eigenvectors are reported as analogous to regression coefficients. The theory of canonical correlations is explained in many multivariate statistics texts and also in Vinod and Ullah (1982). In R, computation of canonical correlations with the function `cancor` is immediately available as a part of the basic stats package.

Economists are often interested in generalizing the regressions to situations where economic theory demands that there be more than one dependent variables. One of the earliest applications is in Vinod (1968, 1976) where joint production is considered. Joint production requires two (or more) dependent variables as outputs besides the usual capital and labor as inputs. Let all columns for the output side variables after subtracting respective means be denoted as Y and all similarly de-meaned columns for input side variables as X. The joint production function is

\[ Y \beta = \rho X \alpha + \epsilon, \]  

where \( \beta \) is a vector of coefficients on the Y (output) side, \( \rho \) is the square root of the largest canonical correlation (eigenvalue) and \( \alpha \) is a vector of coefficients on the X (input) side and \( \epsilon \) represents a vector of errors. Our interest is usually focused only on the first eigenvalue and corresponding eigenvectors on both sides, even though there are as many vectors and coefficients as \( \min(x, y) \), where \( x \) (respectively, \( y \)) denotes the number of variables in the x (resp., y) side of the equation. Hotelling has proved that \( \beta \) and \( \alpha \) vectors (weights) associated with the largest canonical correlation (eigenvalue) is optimal, in the sense that it is best-fitting.

Many econometric problems have a need for joint estimation besides joint production functions. The y-x pairs can also represent (i) jointly dependent variables against exogenous variables, (ii) effect against causal variables, (iii) policy effect and policy variables or (iv) domestic and international variables. One can define \( \gamma = \rho \alpha \) and interpret \( F = Y \beta - X \gamma = 0 \), as an implicit function of Y and X variables and find the partial derivatives of any pair of variables by using the implicit function theorem: For example, if \( y_1 \) is one variable in the set Y and \( x_1 \) is a variable in the set X, we use

\[ \left( \frac{dy_1}{dx_1} \right) = -\frac{\text{Num}}{\text{Den}}, \text{ where } \text{Num} = \partial F/\partial x_1, \text{ Den} = \partial F/\partial y_1. \]  

The derivative \( \left( \frac{dx_1}{dy_1} \right) \) can also be similarly computed when both variables are from the same set. Note that various elasticities of interest to economists can be formulated by using (5.2.2) and we shall give R programs for computing bootstrap estimates of confidence intervals for coefficients. These programs can be readily modified to find confidence intervals on partial derivatives even when some simple types of nonlinearities (such as logs, cross-product or squares of variables) are present in the original specification. All one has to do is to compute a large number (=999, say) of bootstrap estimates of elasticities, rank-order them and finally use the 25-th and 975-th ordered values as 95% confidence limits.
The ‘cancor’ function in R uses economic data called ‘LifeCycleSavings’ to illustrate it. This data is from 1960 to 1970 containing a cross section of 50 countries on 5 variables. The variables are: sr (aggregate personal saving divided by disposable income), pop15 (% of population under 15), pop75 (% of population over 75), dpi (real per-capita disposable income), and ddpi (% rate of change in per-capita disposable income). Modigliani’s famous life-cycle savings model explains ‘sr’ by dpi, ‘ddpi’ and the two demographic variables ‘pop15’ and ‘pop75’. The data are averaged over the decade 1960–1970 to remove the business cycle or other short-term fluctuations. The canonical correlation is between ny=3 endogenous variables (sr, dpi, ddpi) and nx=2 exogenous variables (pop15, pop75). Since minxy=2 here, software will report two sets of coefficients on the x side and y side and two sets of estimates of ρ. However we are interested in only the first set associated with the largest eigenvalue as the best fitting linear combination between endogenous and exogenous variables.

Since eigenvectors are unique up to sign, the arbitrary sign of coefficients can be adjusted by simply making the sign of the weight on the savings ratio (sr) as the basic dependent variable always positive. The R snippet is:

```r
#R5.2.1
attach(LifeCycleSavings) #make names accessible
reg1=lm(sr~dpi+ddpi+pop15+pop75)
# (Intercept)          dpi         ddpi        pop15        pop75
#  28.5660865   -0.0003369    0.4096949   -0.4611931   -1.6914977
# t’s   3.9       -0.36            2.1         -3.2      -1.6
cc=cancor(cbind(pop15,pop75), cbind(sr2,dpi2,ddpi2))
# x first then y is counterintuitive to regressing y on x,
# but this is the way it is in R
cc
alph=cc$xcoef[,1]
bet=cc$ycoef[,1]
if (bet[1]<0) {bet=-bet; alph=-alph}
rho=sqrt(cc$cor[1])
print("The fitted relation is:")
cout=c(bet,rho*alph)
cout
#            sr           dpi          ddpi        pop15        pop75
#  0.0084710221  0.0001307398  0.0041706000 -0.0082743260  0.0441808517
```

**Why canonical correlation is not popular so far?** Even though many econometric problems have multiple output type variables, this method has failed to appeal due to three reasons. (i) The eigenvectors as weights of variables are not unique and are perfectly valid if all signs are changed. (ii) If the variables are correlated with each other due to collinearity, the eigenvectors are unreliable. (iii) No standard errors are available for assessing the individual coefficients although ρ is a good measure of goodness of overall fit among two linear combinations. Regarding first problem, the solution is simply to choose that sign which makes economic sense. Vinod (1976) solves the second problem by using ridge regression. The third problem has ’been solved by assuming asymptotics by researchers including Dhrymes. However a simple practical solution was
not available till the advent of the bootstrap. However, it is now quite a simple matter to create 999 estimates of coefficients and determine confidence intervals from them.

```r
#R5.2.2 bootstrap estimate of standard errors
attach(LifeCycleSavings) #make names accessible
n999=999; nx=2; ny=3; nall=nx+ny
ccoef=matrix(rep(NA, nall*n999), nrow=n999, ncol=nall) #place to store answers
set.seed(234)
for (i in 1:n999){
sr2=sample(sr,replace=T)
dpi2=sample(dpi,replace=T)
pop152=sample(pop15,replace=T)
pp752=sample(pop75,replace=T)
cc=cancor(cbind(pop152,pop752), cbind(sr2,dpi2,pp752)) # x first then y
alph=cc$xcoef[,1]
bet=cc$ycoef[,1]
rho=sqrt(cc$cor[1])
#print("The fitted relation is:")
#if sign of weight on sr is negative, change all signs
if (bet[1]<0) {bet=-bet; alph=-alph}
#print(c(bet,rho*alph))
ccoef[i,]=c(bet,rho*alph)}
for (j in 1:nall){x=sort(cccoef[,j])
ccoef[,j]=x} #sort all columns individually
print("Lower and upper Limit of a 95% confidence interval")
lolim=round(cccoef[round((n999)*0.025),],4)
uplim=round(cccoef[round((n999)*0.975),],4)
nam=c(colnames( cbind(sr2,dpi2,pp752)),colnames(cbind(pop152,pop752)))
print(cbind(nam,lolim,uplim),q=F)
```

Since all confidence intervals contain the zero, except the (artificially adjusted) one with ‘sr’, none of the coefficients are significant. This evidence based on canonical correlations optimal fit rejects Modigliani’s life cycle model of consumer savings behavior. The idea of using the bootstrap for inference in canonical correlations is my original idea, not reported anywhere so far.

If the original data are time series instead of cross section as in the example above one can use the R package called meboot to do the bootstrap. The code below illustrates the estimation of wool and mutton by capital and labor based on the data in Vinod (1969).

```r
#R5.2.3 Joint production of Wool and Mutton
labor=c(17703,18552, 18814, 13490, 13390, 12710, 13100, 13350, 14280, 14710, 13290, 12560)
cap=c(193649, 227548, 206165, 261888, 260120, 262540, 286690, 307320, 309180,304810)
wool=c(228091, 233309, 232258, 235807, 241284, 242177, 239101, 243713, 259939,166563, 261249, 248538)
mutton=c(11416, 14304, 16321, 16255, 16553, 16328, 15292, 14495, 15528, 16239, 17536, 17171)
labor=ts(labor, frequency=1, start=c(1951,1))
```
cap = ts(cap, frequency=1, start=c(1951,1))
wool = ts(wool, frequency=1, start=c(1951,1))
mutton = ts(mutton, frequency=1, start=c(1951,1))
nn = 10000

cap = cap/nn; labor = labor/nn; wool = wool/nn; mutton = mutton/nn

layout(matrix(1:2, nrow = 2, ncol = 1))
plot.ts(wool, main="Production of wool")
plot.ts(mutton, main="Production of mutton")
plot.ts(cap, main="Capital input")
plot.ts(labor, main="Labor input")

cc = cancor(cbind(cap, labor), cbind(wool, mutton))
# x first then y is counterintuitive to regressing y on x,
# but this is the way it is in R

calph = cc$xcoef[,1]
bet = cc$ycoef[,1]
if (bet[1]<0) {bet = -bet; alph = -alph}
rho = sqrt(cc$cor[1])
print("The fitted relation is:")
orig = c(bet, rho*alph)

orig #original coefficients before any bootstrap
#check that we got best fitting linear combination
dewool = wool - mean(wool)
demutt = mutton - mean(mutton)
decap = cap - mean(cap)
delab = labor - mean(labor)

cbind((orig[1]*dewool + orig[2]*demutt), (orig[3]*decap + orig[4]*delab))
mean((orig[1]*dewool + orig[2]*demutt) - (orig[3]*decap + orig[4]*delab))

#average error is zero

n999 = 999; nx = 2; ny = 2; null = nx+ny
cccoef = matrix(rep(NA, nall*n999), nrow=n999, ncol=nall)

library(meboot)
set.seed(234)
wool2 <- meboot(x=wool, reps=n999, trim=0.10, elaps=F)
mutton2 <- meboot(x=mutton, reps=n999, trim=0.10, elaps=F)
cap2 <- meboot(x=cap, reps=n999, trim=0.10, elaps=F)
labor2 <- meboot(x=labor, reps=n999, trim=0.10, elaps=F)

for (i in 1:n999) {
  capz = cap2$ens[,i]; laborz = labor2$ens[,i]
  woolz = wool2$ens[,i]; muttonz = mutton2$ens[,i]
  cc = cancor(cbind(capz, laborz), cbind(woolz, muttonz))
  alph = cc$xcoef[,1]
  bet = cc$ycoef[,1]
  rho = sqrt(cc$cor[1])
  #print("The fitted relation is:")
  if (bet[1]<0) {bet = -bet; alph = -alph}
  #print(c(bet, rho*alph))
  cccoef[i,j] = c(bet, rho*alph) #store the coefficients
}

#end the loop using meboot resamples
#Now find confidence intervals from reincarnations
for (j in 1:nall) {x = sort(cccoef[,j])
  cccoef[,j] = x #sort all columns individually and end j loop
}

print("Lower and upper Limit of a 95% confidence interval")
lolim=round(cccoef[round((n999)*0.025)],4)# lower lim at 2.5%
uplim=round(cccoef[round((n999)*0.975)],4)# upper at 97.5%
nam=c(colnames( cbind(wool,mutton)),colnames(cbind(cap,labor)))
print("Name, Lower, Original, Upper")
orig=round(orig,4)# round original coefficients for printing
print(cbind(nam,lolim,orig,uplim),q=F)# output follows
# nam  lolim  orig  uplim
#[1,] wool  0.0127  0.0312  0.3352
#[2,] mutton -2.2689 1.7236  2.8714
#[3,] cap    -0.041  0.0119  0.0375
#[4,] labor  -1.9498 -0.8472 0.0613

Production of wool

Production of mutton
The results suggest that labor input has a negative impact on the output of wool and mutton. However, the analysis of confidence intervals clarifies that the labor coefficient is not statistically significant since the ‘meboot’ confidence interval contains the zero. The negative effect of labor in aggregate data may have to do with labor-saving technological changes. It is clear from the graphics that the dips and hills in the four series appear to be unrelated to each other and this is confirmed by statistically insignificant results. We need better data and better specification which would allow for inventories, time lags, effect of weather on demand for wool, effect of prices of fish, chicken, pork and beef, incomes, etc.

R5.2.4 Joint production of Wool and Mutton Cobb-Douglas specification
labor=c(17703, 18552, 18814, 13490, 13390, 12710, 13100, 13350, 14280, 14710, 13290, 12560)
cap=c(193649, 227548, 206165, 261888, 260120, 158990, 262540, 286690, 307320, 309180, 304810)
5.3 VAR Estimation and testing of Canonical Correlations.

Johansen and coauthors in a series of papers since 1990’s have developed some advanced tools for unit root testing and cointegration estimation. Cointegration was discussed in Section 3.4. Many of these are available in R package called ‘urca’ by Bernhard Pfaff. Following functions and datasets in the ‘urca’ package are useful except that in most cases the functions do not end with a list command so that all the outputs are simply printed, but not created as objects available for further manipulation, collection, tabulation, etc. A better alternative is to use the ‘tseries’ package for some of these functions. The following list includes mostly urca package functions.

The VAR(p) model of (5.1.1) is written in error correction model (ECM) form:

\[ \Delta X_t = \Sigma_{i=1}^{p-1} \Gamma_i \Delta X_{t-i} + \Pi X_{t-p} + \Phi D_t + \mu + \epsilon_t, \quad (t=1, \ldots, T), \]  

(5.3.1)
where \( D_t \) are seasonal dummies orthogonal to the constant term. The ECM writes \( \Delta X_t \) as a function of lagged levels \( X_{t-1} \) and lagged \( \Delta X_t \). The name error correction arises because lagged \( \Delta X_t \) can be viewed as past equilibrium errors. It is interesting that under certain conditions one need not actually estimate (5.3.1) but simply re-use the lag coefficient matrices of \( A_1 \) to \( A_p \) in (5.1.1) and recover the \( \Pi \) and \( \Gamma \) matrices as follows:

\[
\Pi = -I_K + A_1 + A_2 + \ldots + A_p, \\
\Gamma_1 = -(A_{i+1} + \ldots A_p).
\]  

(5.3.2)

The estimation of VAR(\( p \)) can be a simple matter of \( p \) OLS regressions. In addition to the OLS, the literature discusses various estimation methods including Engle-Granger two-step, feasible generalized least squares (FGLS), and maximum likelihood (ML). Of course, we need \( \varepsilon \) in (5.3.1) to be Gaussian errors with zero mean and covariance matrix \( \Lambda \) to write the likelihood function, usually written conditional on the first \( p \) data points, assumed to be fixed (not random).

The cointegration hypothesis (see Sec. 3.4) becomes a restriction on the matrix \( \Pi \) matrix (coefficients of lagged \( X \) variables). The model in (5.3.1) is the hypothesis \( H_1 \) and hypothesis of at most \( r \) cointegrating vectors is stated as a decomposition of the \( \Pi \) matrix:

\[ H_2: \quad \Pi = \alpha \beta', \]  

(5.3.3)

where \( \beta \) contains the cointegrating vectors and \( \alpha \) contains the adjustment coefficients, both of which are \( p \times r \) matrices (\( p > r \)). Johansen explains that the matrices of \( \alpha \) and \( \beta \) cannot be uniquely known, that is, they are not identified. The space spanned by \( \beta \) is the row space and the space spanned by \( \alpha \) is the column space of \( \Pi \). The hypothesis of cointegration is the hypothesis of reduced rank of \( \Pi \) and a direct way of checking it is to see if any of its singular values are close to zero. Johansen shows that maximum likelihood estimation amounts to calculating the singular values of a matrix sandwiched so that the estimate \( \Pi \) of \( \Pi \) is in the middle. If the rank of \( \Pi \) is \( r \), we have \( r \) cointegrating vectors.

Once the notation two matrices \( \alpha \) and \( \beta \) appearing in (5.3.3) is understood the following tests in the urca package of R can be used for various cointegration testing tasks.

\begin{verbatim}
ablrtest =Likelihood ratio test for restrictions on alpha and beta
Ukpppuip = Data set for the United Kingdom: ppp and uip
alphaols =OLS regression of VECM (vector error correction) weighting matrix
alrtest =Likelihood ratio test for restrictions on alpha
bh5lrtest =Likelihood ratio test for restrictions under partly known beta
blrtest =Likelihood ratio test for restrictions on beta
ca.jo =Johansen Procedure for VAR, reports trace
ca.po =Phillips & Ouliaris Cointegration Test
cajolst =Testing Cointegrating Rank with Level Shift at Unknown time
cajools =OLS regression of VECM
lttest =Likelihood ratio test for no linear trend in VAR
plotres =Graphical inspection of VECM residuals
\end{verbatim}
ur.df = Augmented-Dickey-Fuller Unit Root Test (No p-values and output cannot be extracted for further manipulation or tabulation as is possible with ‘adf.test’ function in package ‘tseries’)