GABRIELE CANTALUPPI

COMPUTATIONAL LABORATORY FOR ECONOMICS

Notes for the students

Addendum 3rd edition

EDUCatt

B

Addendum 3rd edition

The R code for some Illustrations presented in the 3rd edition of Verbeek's A Guide to Modern Econometrics are reported here.

B.1 Annual Price/Earnings Ratio (Section 8.4.4 third edition)

Data on the ratio of the S&P composite stock price index and S&P composite earnings over the period 1871–2002 ($T = 132$) are considered; they can be read by means of the function `read.table`, having extracted the file `pe.dat` from the compressed archive `price-earnings_ratio.zip`.
The function `ts(object, start, frequency)` creates a multiple time series from the columns of a table; in this case there is no need to specify the frequency since data are annual.

```
> pe <- read.table(unzip("price-earnings_ratio.zip",
        "pe.dat"), header = TRUE)
> pe <- ts(pe, start = 1871)
```

The following variables are available:

- `price` S&P composite stock price index
- `earnings` S&P composite earnings index
- `pe` price/earnings
- `logpe` log(pe)

To obtain a plot of the log of the price/earnings series use the function `xyplot` available in the package `lattice`. Remind that a multiple time series, `mts`, object can be treated like a matrix so it is possible to make reference to the third column of the object `pe`, containing the price/earnings ratio, and apply the logarithm function, or make direct reference to the fourth column of `pe` (`xyplot(pe[,4])`).

```
> library(lattice)
> xyplot(log(pe[, 3]))
```

B.1.1 Dickey-Fuller test

The function `ur.df` available in the package `urca` computes the Dickey-Fuller test (see Section 8.7.1 for the Dickey-Fuller test construction). The function `ur.df` has

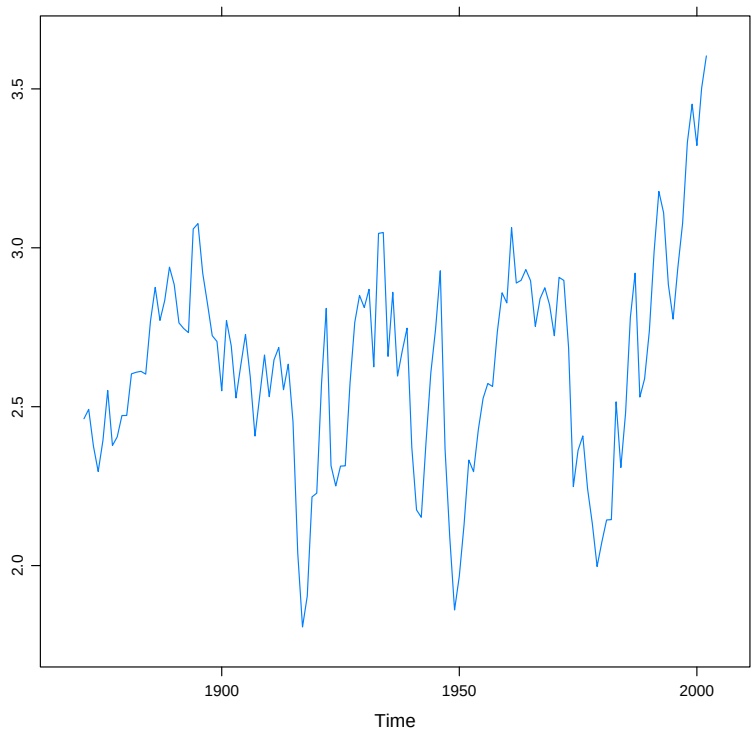

Figure B.1 Annual log price/earnings ratio, 1871–2002

four parameters. The first is a time series. With the parameter `type` it is possible to specify if only a constant has to be included in model (8.12) (`type="drift"`) or both a trend and the drift (`type="trend"`) or neither the drift nor the trend (`type="none"`) have to be included. The parameter `lags` specifies a number of lags for Y_t to include in the regression (**??**); `selectlags`, which by default is equal to `"fixed"`, may be set to `"AIC"` or `"BIC"` for obtaining an automatic lag selection according to the Akaike or the Bayesian Information criteria, within the maximum number of lags specified by `lags`.

```
> library(urca)
> y <- log(pe[, 3])
> summary(ur.df(y, type = "drift", lags = 0))
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift
```

```
Call:
lm(formula = z.diff ~ z.lag.1 + 1)

Residuals:
     Min       1Q   Median       3Q      Max
-0.53078 -0.10211  0.00324  0.12171  0.41194

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.33486    0.12790   2.618   0.0099 **
z.lag.1     -0.12452    0.04847  -2.569   0.0113 *
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.1777 on 129 degrees of freedom
Multiple R-squared:  0.04867,        Adjusted R-squared:  0.04129
F-statistic:   6.6 on 1 and 129 DF,  p-value: 0.01134


Value of test-statistic is: -2.569 3.4573

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
```

The resulting statistic (the first value in the section `Value of test-statistic is:`) is the augmented Dickey-Fuller Test for unit root.
Verbeek provides tests for up to six additional lags of ΔY_t. These can be obtained by creating an appropriate function. Observe that the function `ur.df` uses classes of type S4, so one has to use the `@`, and not the $ sign, to extract a slot[1] from an object of class S4 produced by `ur.df`, see `str(ur.df(y,type="drift",lags=6))`.

```
> f <- function(x) {
        urtest <- ur.df(y, type = "drift", lags = x)
        c(stat = urtest@teststat[1], "5% crit. value" = urtest@cval[1,
            2])
    }
> a <- 0:6
> names(a) <- c("DF", paste("ADF(", 1:6, ")", sep = ""))
> round(sapply(a, f), 3)
                   DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
stat           -2.569 -2.999 -2.487 -2.503 -1.778 -1.627 -1.825
5% crit. value -2.880 -2.880 -2.880 -2.880 -2.880 -2.880 -2.880
```

[1] Elements of objects belonging to an S4 class are named slots.

The function `f` with argument `x` is defined, which extracts from the object resulting from `ur.df` applied to the time series `y`, with `type="drift"` and `lags=x`, the first element of the slot `teststat`, which is the Dickey-Fuller statistic, and the element in the first row, second column of the slot `cval`, which is the 5% critical value (see also the section `Critical values for test statistics` in the preceding outputs).
The variable `a` contains the desired lags for the unit root test.
The names `DF` and `ADF(1)` to `ADF(6)` are assigned to the elements of `a`.
The function `sapply` is finally used to call the function `f` for the different values of the lags in the array `a`.

B.1.2 Testing for multiple unitary roots

By imposing a first unit root it is possible to test for the presence of a second unit root with regressions of the form:

$$\Delta^2 Y_t = \delta + \pi \Delta Y_{t-1} + c_1 \Delta^2 Y_{t-1} + \cdots + e_t \tag{B.1}$$

the null hypothesis corresponds to $\pi = 0$.
We can define a function `g` computing the Dickey-Fuller statistic for the differenced series `diff(y)` and then use the function `sapply` to obtain the results of the function `g` applied to the different values of the lags in the array `a`, defined above. (The 5% critical value is the same as before).

```
> g <- function(x) summary(ur.df(diff(y), type = "drift",
        lags = x))@teststat[1]
> (adf <- round(sapply(a, g), 3))
     DF  ADF(1)  ADF(2)  ADF(3)  ADF(4)  ADF(5)  ADF(6)
-10.588  -9.109  -7.310  -7.481  -6.431  -5.315  -4.210
```

The plot for the annual change in log price/earnings ratio (first differenced series) can be obtained by applying the function `diff` to the log series of the price/earnings ratio.

```
> library(lattice)
> xyplot(diff(log(pe[, 3])))
```

B.2 Modelling the Price/Earnings Ratio (Section 8.7.5 third edition)

Make reference to Section B.1 for data reading and `mts` creation.

```
> pe <- read.table(unzip("price-earnings_ratio.zip",
        "pe.dat"), header = TRUE)
> pe <- ts(pe, start = 1871)
```

In Section B.1 the hypothesis of a unit root in the log of the price/earnings series wasn't rejected; we now present Verbeek's description to model the first-differenced series.
Let `Dy` be the log differenced series.

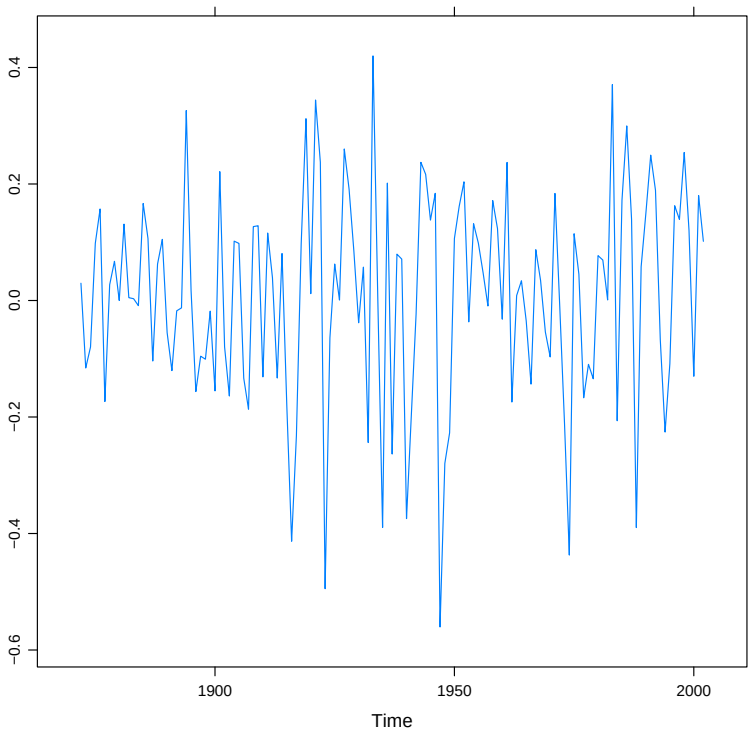

Figure B.2 Annual change in log price/earnings ratio, 1872–2002

```
> Dy <- diff(pe[, 4])
```

From the analysis of the autocorrelation function and of the partial autocorrelation function, see Fig. B.3, which can be obtained with the code:

```
> library(astsa)
> t(acf2(Dy, 20))
     [,1]  [,2]  [,3]  [,4]  [,5] [,6] [,7]  [,8]  [,9] [,10]
ACF  0.07 -0.17 -0.06 -0.18 -0.05 0.09 0.13 -0.01 -0.02  0.06
PACF 0.07 -0.17 -0.04 -0.21 -0.05 0.02 0.09 -0.04  0.01  0.09
     [,11] [,12] [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20]
ACF  -0.08 -0.02 -0.10 -0.05  0.04  0.01  0.03 -0.02 -0.04 -0.08
PACF -0.04  0.02 -0.15 -0.03 -0.02 -0.03 -0.02 -0.04 -0.02 -0.07
```

it emerges that autocorrelations and partial autocorrelations are zero from lag 4. Observe that in the package `FitAR`, the function `PacfPlot` is available, which allows a graphical representation of the 95% confidence intervals for the partial autocorrelations to be obtained, see Fig. B.4.

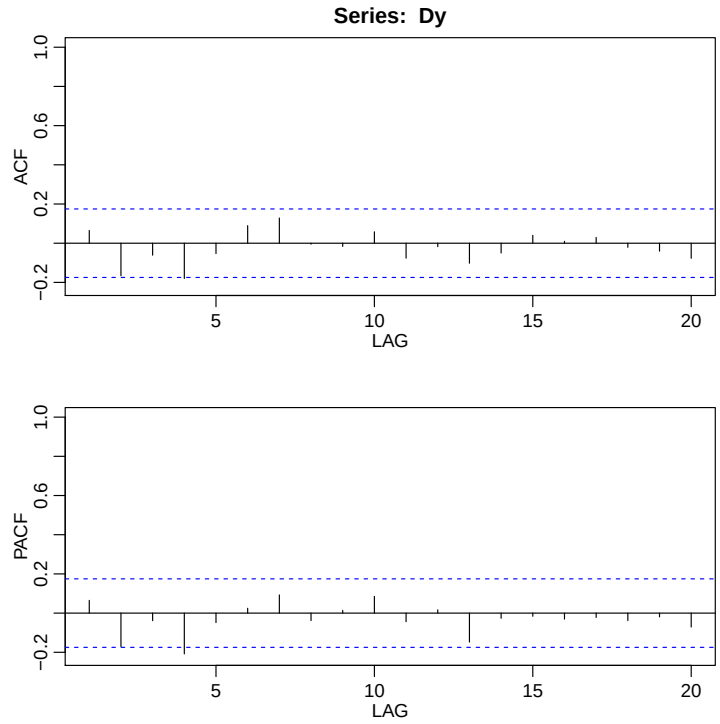

Figure B.3 Sample autocorrelation and partial autocorrelation functions of $\Delta \log(P/E)$

```
> library(FitAR)
> PacfPlot(Dy, lag.max = 20)
```

Verbeek initially proposes the estimation of an AR(4) and an MA(4) model:

$$\Delta Y_t = \delta + \theta_1 \Delta Y_{t-1} + \theta_2 \Delta Y_{t-2} + \theta_3 \Delta Y_{t-3} + \theta_4 \Delta Y_{t-4} + \varepsilon_t$$

and

$$\Delta Y_t = \delta + \varepsilon_t + \alpha_1 \varepsilon_{t-1} + \alpha_2 \varepsilon_{t-2} + \alpha_3 \varepsilon_{t-3} + \alpha_4 \varepsilon_{t-4} \tag{B.2}$$

B.2.1 AR estimation

We can use the function `dynlm` in the package `dynlm` to obtain the estimate of the AR(4) model by OLS :

```
> library(dynlm)
> ar4regr <- dynlm(Dy ~ L(Dy) + L(Dy, 2) + L(Dy, 3) +
        L(Dy, 4))
> summary(ar4regr)
```

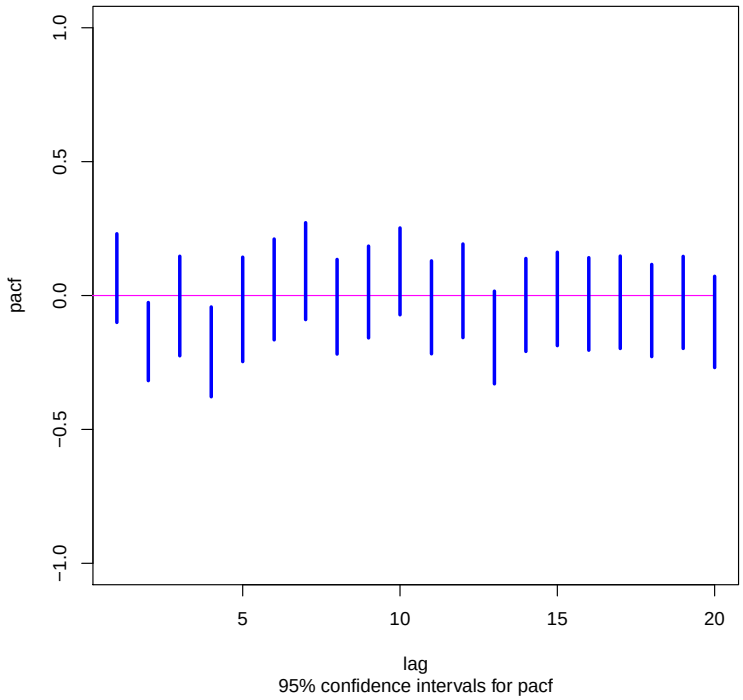

Figure B.4 95% confidence interval for the sample partial autocorrelation function of $\Delta \log(P/E)$

```
Time series regression with "ts" data:
Start = 1876, End = 2002

Call:
dynlm(formula = Dy ~ L(Dy) + L(Dy, 2) + L(Dy, 3) + L(Dy, 4))

Residuals:
     Min       1Q   Median       3Q      Max
-0.50034 -0.10358  0.01051  0.12173  0.45118

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01209    0.01598   0.757   0.4506
L(Dy)        0.06021    0.08849   0.680   0.4975
L(Dy, 2)    -0.20312    0.08898  -2.283   0.0242 *
L(Dy, 3)    -0.02341    0.08899  -0.263   0.7930
```

```
L(Dy, 4)    -0.21165    0.08906  -2.377   0.0190 *
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.1793 on 122 degrees of freedom
Multiple R-squared:  0.07706,        Adjusted R-squared:  0.0468
F-statistic: 2.547 on 4 and 122 DF,  p-value: 0.04281
```

The code `ar4regr<-dynlm(Dy L(Dy,1:4))` gives the same result.
The estimate of the mean of the process can be obtained as:

```
> mean(Dy)
[1] 0.008715682
```

or in an indirect way by using the estimate of the intercept and of the autoregressive parameters:

```
> ar4regr$coef[[1]]/(1 - sum(ar4regr$coef[-1]))
[1] 0.008775719
```

Estimation via maximum likelihood can be obtained with the function `arima`, see Section 8.5.

B.2.2 The Ljung-Box statistic

The Ljung-Box statistics, see Section 8.10.2, for the first 6 autocorrelations can be obtained by making use of the function `Box.test(x, lag, type, fitdf)`, where `x` is a univariate time series object; `lag` specifies the number of lags to consider; `type` the statistic to compute `"Box-Pierce"` or `"Ljung-Box"`; `fitdf` is the number of degrees of freedom to be subtracted to `lag`, if `x` is a series of residuals, usually `fitdf = p+q` (where `p` and `q` are the orders respectively of the AR and MA parts of an ARMA model describing the process level) so the degrees fo freedom are $\texttt{lag} - (\texttt{p} + \texttt{q})$, provided of course that `lag > fitdf`.
The Ljung-Box statistics for the first 6 and 12 autocorrelations result:

```
> Box.test(residuals(ar4regr), lag = 6, type = "Ljung-Box",
        fitdf = 4)
        Box-Ljung test

data:  residuals(ar4regr)
X-squared = 0.2552, df = 2, p-value = 0.8802
> Box.test(residuals(ar4regr), lag = 12, type = "Ljung-Box",
        fitdf = 4)
        Box-Ljung test

data:  residuals(ar4regr)
X-squared = 3.6131, df = 8, p-value = 0.8902
```

To obtain AIC and BIC according to Verbeek formulae (8.68) and (8.69) use

```
> T = length(Dy)
> log(summary(ar4regr)$sigma^2) + 2 * (4 + 1)/T
[1] -3.360744
> log(summary(ar4regr)$sigma^2) + (4 + 1)/T * log(T)
[1] -3.251003
```

B.2.3 AR estimation via Maximum Likelihood

We can use the function `arima` to obtain the estimate of the AR(4) model:

```
> (ar4est <- arima(Dy, c(4, 0, 0)))
Call:
arima(x = Dy, order = c(4, 0, 0))

Coefficients:
         ar1      ar2      ar3      ar4  intercept
      0.0601  -0.2029  -0.0228  -0.2067     0.0084
s.e.  0.0852   0.0857   0.0854   0.0851     0.0111

sigma^2 estimated as 0.03015:  log likelihood = 43.35,  aic = -74.71
```

B.2.4 MA estimation

With reference to relationship (B.2):

$$\Delta Y_t = \delta + \varepsilon_t + \alpha_1\varepsilon_{t-1} + \alpha_2\varepsilon_{t-2} + \alpha_3\varepsilon_{t-3} + \alpha_4\varepsilon_{t-4}$$

to obtain OLS estimates we can use the function `arima` with `method="CSS"` which minimizes the Conditional Sum of Squares, see `?arima`.
Observe that the mean is returned named as `intercept`.

```
> (ma4est <- arima(Dy, c(0, 0, 4), method = "CSS"))
Call:
arima(x = Dy, order = c(0, 0, 4), method = "CSS")

Coefficients:
         ma1      ma2      ma3      ma4  intercept
      0.0473  -0.1895  -0.0410  -0.1503     0.0080
s.e.  0.0866   0.0917   0.0828   0.0930     0.0103

sigma^2 estimated as 0.03047:  part log likelihood = 42.78
```

and the Ljung-Box tests for the first 6 and 12 autocorrelations result

```
> sapply(c(6, 12), function(i) Box.test(residuals(ma4est),
        lag = i, type = "Ljung-Box", fitdf = 4))
           [,1]                [,2]
statistic 1.537293            5.013894
parameter 2                   8
p.value   0.4636402           0.7560901
method    "Box-Ljung test"    "Box-Ljung test"
data.name "residuals(ma4est)" "residuals(ma4est)"
```

With maximum likelihood we would obtain:

```
> (ma4est <- arima(Dy, c(0, 0, 4)))
Call:
arima(x = Dy, order = c(0, 0, 4))

Coefficients:
         ma1      ma2      ma3      ma4  intercept
      0.0479  -0.1875  -0.0401  -0.1462     0.0080
s.e.  0.0865   0.0913   0.0819   0.0916     0.0104

sigma^2 estimated as 0.03047:  log likelihood = 42.69,  aic = -73.39
```

B.2.5 Non complete models

Finally Verbeek suggests the estimation of non-complete models, including for the AR only the second and fourth lags. Namely, by examining the output of the AR(4) model we note that only the second and fourth lags are significant.

The estimation of subsets ARMA models can be obtained by using the argument `fixed` in the `arima` function. Only the parameter corresponding to `NA` in the vector, which must have the same length of the number of the parameters, will be estimated. Ljung-Box statistics are also reported for lags 6 and 12 (The number of degrees of freedom is named `parameter`).

OLS parameter estimates can be obtained by using the function `dynlm`.

```
> summary(dynlm(Dy ~ L(Dy, 2) + L(Dy, 4)))
Time series regression with "ts" data:
Start = 1876, End = 2002

Call:
dynlm(formula = Dy ~ L(Dy, 2) + L(Dy, 4))

Residuals:
     Min       1Q   Median       3Q      Max
-0.49363 -0.10716  0.00548  0.11819  0.43736

Coefficients:
```

```
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.01252    0.01585   0.790   0.4313
L(Dy, 2)    -0.20201    0.08800  -2.296   0.0234 *
L(Dy, 4)    -0.21703    0.08817  -2.462   0.0152 *
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.1783 on 124 degrees of freedom
Multiple R-squared:  0.07246,        Adjusted R-squared:  0.0575
F-statistic: 4.843 on 2 and 124 DF,  p-value: 0.009433
```

The model can also be estimated with maximum likelihood

```
> (ar4restr <- arima(Dy, c(4, 0, 0), fixed = c(0, NA,
        0, NA, NA)))
Call:
arima(x = Dy, order = c(4, 0, 0), fixed = c(0, NA, 0, NA, NA))

Coefficients:
      ar1      ar2  ar3      ar4  intercept
        0  -0.2016    0  -0.2119     0.0084
s.e.    0   0.0854    0   0.0849     0.0108

sigma^2 estimated as 0.0303:  log likelihood = 43.03,  aic = -78.06
> sapply(c(6, 12), function(i) Box.test(residuals(ar4restr),
        lag = i, type = "Ljung-Box", fitdf = 4))
          [,1]                 [,2]
statistic 0.7285699            4.05422
parameter 2                    8
p.value   0.6946932            0.8521987
method    "Box-Ljung test"     "Box-Ljung test"
data.name "residuals(ar4restr)" "residuals(ar4restr)"
```

With regard to the estimation of a subset MA(4) model including only the second and fourth order coefficients we have:

```
> (ma4restr <- arima(Dy, c(0, 0, 4), fixed = c(0, NA,
        0, NA, NA)))
Call:
arima(x = Dy, order = c(0, 0, 4), fixed = c(0, NA, 0, NA, NA))

Coefficients:
      ma1      ma2  ma3      ma4  intercept
        0  -0.1810    0  -0.1563     0.0080
s.e.    0   0.0911    0   0.0923     0.0102

sigma^2 estimated as 0.03057:  log likelihood = 42.47,  aic = -76.94
```

```
> sapply(c(6, 12), function(i) Box.test(residuals(ma4restr),
        lag = i, type = "Ljung-Box", fitdf = 4))
          [,1]                 [,2]
statistic 2.219822             5.480006
parameter 2                    8
p.value   0.3295883            0.7052541
method    "Box-Ljung test"     "Box-Ljung test"
data.name "residuals(ma4restr)" "residuals(ma4restr)"
```

Then Verbeek suggests to estimate[2]:

- an AR(2) model with only the second lag.

```
> (ar2restr <- arima(Dy, c(2, 0, 0), fixed = c(0, NA,
        NA)))

Call:
arima(x = Dy, order = c(2, 0, 0), fixed = c(0, NA, NA))

Coefficients:
      ar1      ar2  intercept
        0  -0.1657     0.0085
s.e.    0   0.0860     0.0134

sigma^2 estimated as 0.03178:  log likelihood = 40,  aic = -73.99

> sapply(c(6, 12), function(i) Box.test(residuals(ar2restr),
        lag = i, type = "Ljung-Box", fitdf = 2))
          [,1]                 [,2]
statistic 7.871311             12.12877
parameter 4                    10
p.value   0.09640763           0.2765308
method    "Box-Ljung test"     "Box-Ljung test"
data.name "residuals(ar2restr)" "residuals(ar2restr)"
```

- a MA(2) with only the second order coefficient:

```
> (ma2restr <- arima(Dy, c(0, 0, 2), fixed = c(0, NA,
        NA)))

Call:
arima(x = Dy, order = c(0, 0, 2), fixed = c(0, NA, NA))

Coefficients:
      ma1      ma2  intercept
        0  -0.2497     0.0084
```

[2] We use maximum likelihood to obtain the parameter estimates.

```
s.e.    0   0.0968     0.0117

sigma^2 estimated as 0.03126:  log likelihood = 41.04,  aic = -76.07

> sapply(c(6, 12), function(i) Box.test(residuals(ma2restr),
        lag = i, type = "Ljung-Box", fitdf = 2))

          [,1]                 [,2]
statistic 5.166903             9.31273
parameter 4                    10
p.value   0.2705968            0.5027038
method    "Box-Ljung test"     "Box-Ljung test"
data.name "residuals(ma2restr)" "residuals(ma2restr)"
```

Verbeek suggests to compare the AIC and BIC criteria to select the best model[3]. To compare AIC and BIC values according to Verbeek formulae (8.68) and (8.69) we can first resume in the vector `var.ests` the values of the estimate of the variances of the errors for the models we have considered and in the vector `penalties` the number of parameters (included the intercept) estimated in each model.

```
> var.ests <- c(ar4est = ar4est$sigma, ma4est = ma4est$sigma,
        ar4restr = ar4restr$sigma, ma4restr = ma4restr$sigma,
        ar2restr = ar2restr$sigma, ma2restr = ma2restr$sigma)
> penalties <- c(4, 4, 2, 2, 1, 1) + 1
```

Then we can compute the AIC and BIC criteria, and search for their respective minima.

```
> T = length(Dy)
> (AIC <- log(var.ests) + 2 * penalties/T)
   ar4est    ma4est  ar4restr  ma4restr  ar2restr  ma2restr
-3.425288 -3.414813 -3.450831 -3.441951 -3.418407 -3.434833
> names(which.min(AIC))
[1] "ar4restr"
> (BIC <- log(var.ests) + penalties/T * log(T))
   ar4est    ma4est  ar4restr  ma4restr  ar2restr  ma2restr
-3.315548 -3.305072 -3.384986 -3.376106 -3.374511 -3.390937
> names(which.min(BIC))
[1] "ma2restr"
```

The identification of the best subset model, via the BIC criterion, can also be performed by having recourse to the function `armasubsets` available in the package `TSA`, see Fig. B.5. Observe that the ARMA solution depends on the maximum order of the AR and MA parts.

[3]The same procedure is realized by `armasubsets` and `forecast::auto.arima` by selecting within a set of feasible models.

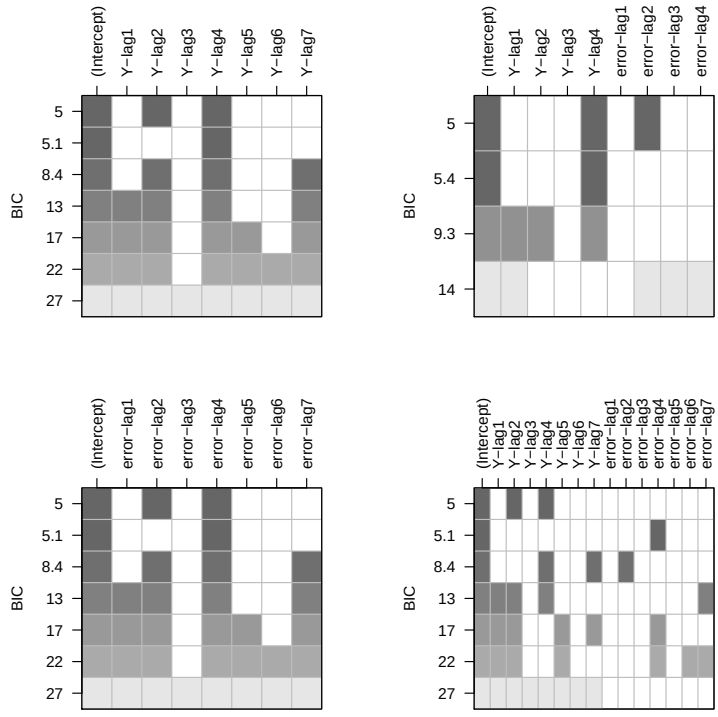

Figure B.5 Identification of non complete models by means of `armasubsets`

```
> library(TSA)
> layout(matrix(1:4, 2, 2))
> plot(armasubsets(Dy, nar = 7, nma = 0))
> plot(armasubsets(Dy, nar = 0, nma = 7))
> plot(armasubsets(Dy, nar = 4, nma = 4))
> plot(armasubsets(Dy, nar = 7, nma = 7))
> detach("package:TSA")
```

We finally apply the procedure `auto.arima` available in the package `forecast`. We use the argument `stepwise=FALSE` for exploring the whole identification structure possibilities:

```
> library(forecast)
> auto.arima(Dy, max.p = 7, max.q = 7, stepwise = FALSE)
Series: Dy
ARIMA(0,0,2) with zero mean

Coefficients:
```

```
          ma1      ma2
      0.0571  -0.2368
s.e.  0.0885   0.0986

sigma^2 estimated as 0.03129:  log likelihood=40.99
AIC=-75.98   AICc=-75.79   BIC=-67.36
```

The procedure suggests a complete ARMA(0,0,2) that is a MA(2) model, since it does not deal with restricted models.

B.3 Volatility in Daily Exchange Rates (Section 8.10.3 third edition)

Data can be read by means of the function `read.table`, having exctracted the file `garch.dat` from the compressed archive `daily_exchange_rates.zip`.

```
> crates <- read.table(unzip("daily_exchange_rates.zip",
        "garch.dat"), header = TRUE)
```

The files garch contain 1867 daily observations on exchange rates of the US dollar against the Deutsch Mark, Canadian Dollar, Japan Yen and Swiss Franc from January 1980 to May, 21, 1987.
The data are irregular because no observations are available for weekends and holidays.
The following variables are available:

- `date` date of observations (yymmdd)
- `day` day of the week (1=monday)
- `dm` exchange rate US$/Deutsche Mark
- `ddm` dm-dm(-1)
- `bp` exchange rate US$/British pound
- `cd` exchange rate US$/Canadian dollar
- `dy` exchange rate US$/Yen
- `sf` exchange rate US$/Swiss Franc

Since data are irregular it is preferable to create an undated time series. The differenced US$/Deutsche Mark exchange rate series is analyzed.

```
> yt <- as.ts(diff(log(crates[, 3])))
```

The graphical representation can be obtained as:

```
> library(lattice)
> xyplot(yt)
```

Verbeek suggests to improve numerical optimization to multiply by 100 the series and obtains the residuals `et` by regressing the series `yt` on a constant.

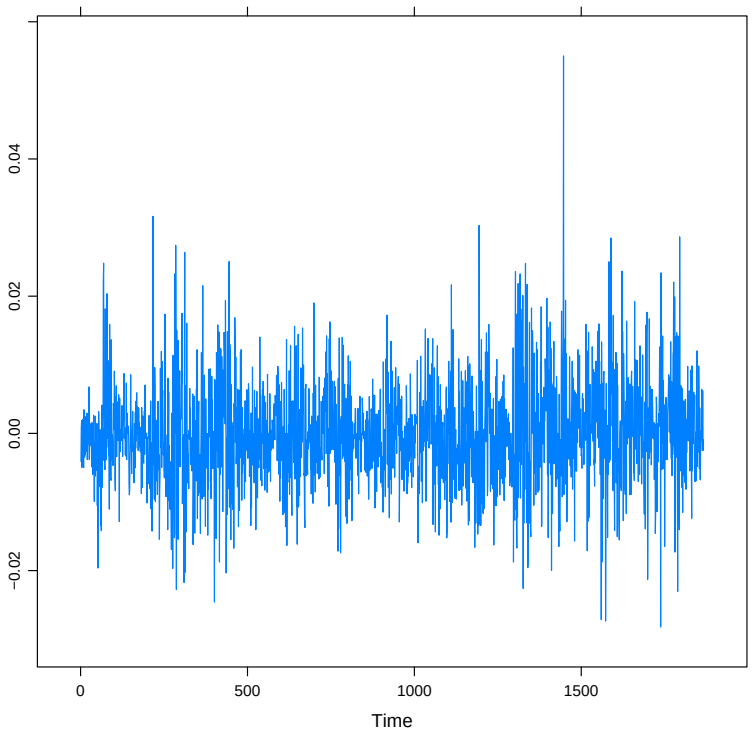

Figure B.6 Daily change in log exchange rate US$/DM, 2 January 1980-21 May 1987

```
> yt <- yt * 100
> library(dynlm)
> et <- dynlm(yt ~ 1)$res
```

Then tests for ARCH effects are performed by regressing the squared residuals, later on `et2`, on a constant and `p` lagged squared residuals series.

```
> et2 <- et^2
> (etsumm <- summary(dynlm(et2 ~ L(et2, 1))))
Time series regression with "ts" data:
Start = 2, End = 1866

Call:
dynlm(formula = et2 ~ L(et2, 1))

Residuals:
    Min      1Q  Median      3Q     Max
-3.0204 -0.5292 -0.3955  0.0534 29.4631
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.53824    0.03179   16.93  < 2e-16 ***
L(et2, 1)    0.10803    0.02303    4.69 2.93e-06 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 1.234 on 1863 degrees of freedom
Multiple R-squared:  0.01167,        Adjusted R-squared:  0.01114
F-statistic:    22 on 1 and 1863 DF,  p-value: 2.925e-06
> etsumm$r.squared * (length(et2) - 1)
[1] 21.76617
> qchisq(0.95, 1)
[1] 3.841459
> (etsumm <- summary(dynlm(et2 ~ L(et2, 1:6))))
Time series regression with "ts" data:
Start = 7, End = 1866

Call:
dynlm(formula = et2 ~ L(et2, 1:6))

Residuals:
    Min      1Q  Median      3Q     Max
-2.7542 -0.4705 -0.3347  0.0871 29.1880

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.35808    0.03972   9.015  < 2e-16 ***
L(et2, 1:6)1  0.07798    0.02304   3.384 0.000729 ***
L(et2, 1:6)2  0.05493    0.02306   2.382 0.017336 *
L(et2, 1:6)3  0.03858    0.02308   1.672 0.094684 .
L(et2, 1:6)4  0.04295    0.02308   1.861 0.062890 .
L(et2, 1:6)5  0.06661    0.02306   2.888 0.003917 **
L(et2, 1:6)6  0.12686    0.02304   5.506 4.19e-08 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 1.217 on 1853 degrees of freedom
Multiple R-squared:  0.04487,        Adjusted R-squared:  0.04178
F-statistic: 14.51 on 6 and 1853 DF,  p-value: 3.063e-16
> etsumm$r.squared * (length(et2) - 6)
[1] 83.45641
> qchisq(0.95, 6)
[1] 12.59159
```

The tests, that can be obtained also by means of the function `ArchTest` in the package `FinTS`, reject the hypothesis of homoscedasticity.

```
> library(FinTS)
> ArchTest(et, lags = 1, demean = FALSE)
        ARCH LM-test; Null hypothesis: no ARCH effects

data:  et
Chi-squared = 21.7662, df = 1, p-value = 3.08e-06
> ArchTest(et, lags = 6, demean = FALSE)
        ARCH LM-test; Null hypothesis: no ARCH effects

data:  et
Chi-squared = 83.4564, df = 6, p-value = 6.661e-16
```

An ARCH(6), a GARCH(1,1) and an EGARCH(1,1) model are estimated. The parameter estimates of the first two models can be obtained by using the function `garchFit` available in the package `fGarch`.

```
> library(fGarch)
> arch6 <- garchFit(formula = ~garch(6, 0), data = as.vector(et),
        include.mean = FALSE, trace = FALSE)
> summary(arch6)
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(6,0),data=as.vector(et),include.mean=FALSE,
    trace = FALSE)

Mean and Variance Equation:
 data ~ garch(6, 0)
<environment: 0x000000001987d500>
 [data = as.vector(et)]

Conditional Distribution:
 norm

Coefficient(s):
   omega    alpha1    alpha2    alpha3    alpha4    alpha5
0.228224  0.089677  0.078667  0.123985  0.139217  0.122020
  alpha6
0.100762

Std. Errors:
 based on Hessian
```

```
Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
omega    0.22822     0.02367    9.643  < 2e-16 ***
alpha1   0.08968     0.02731    3.283  0.00103 **
alpha2   0.07867     0.02590    3.038  0.00238 **
alpha3   0.12399     0.03106    3.991 6.57e-05 ***
alpha4   0.13922     0.03432    4.057 4.98e-05 ***
alpha5   0.12202     0.02996    4.073 4.64e-05 ***
alpha6   0.10076     0.02966    3.398  0.00068 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Log Likelihood:
 -2080.043    normalized:  -1.114707

Description:
 Fri May 24 17:26:50 2013 by user: gabriele.cantaluppi


Standardised Residuals Tests:
                               Statistic p-Value
 Jarque-Bera Test   R    Chi^2  123.0277  0
 Shapiro-Wilk Test  R    W      0.9901219 5.864397e-10
 Ljung-Box Test     R    Q(10)  24.20371  0.007077661
 Ljung-Box Test     R    Q(15)  26.10378  0.03694588
 Ljung-Box Test     R    Q(20)  32.92181  0.03441329
 Ljung-Box Test     R^2  Q(10)  12.70312  0.2407474
 Ljung-Box Test     R^2  Q(15)  24.19968  0.06177924
 Ljung-Box Test     R^2  Q(20)  28.59124  0.0961256
 LM Arch Test       R    TR^2   14.1484   0.2913326

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
2.236917 2.257667 2.236889 2.244563
```

The function `plot` applied to an `fGARCH` object deriving from a `garchFit` estimate, produces the graphs in Fig. B.7. See the help system for a description of the graphs.

```
> layout(matrix(1:12, 4, 3))
> plot(arch6, which = 1:12)
```

The function `predict` allows to obtain forecasts.

```
> predict(arch6, 4)
  meanForecast meanError standardDeviation
1            0 0.5814766         0.5814766
2            0 0.5770533         0.5770533
```

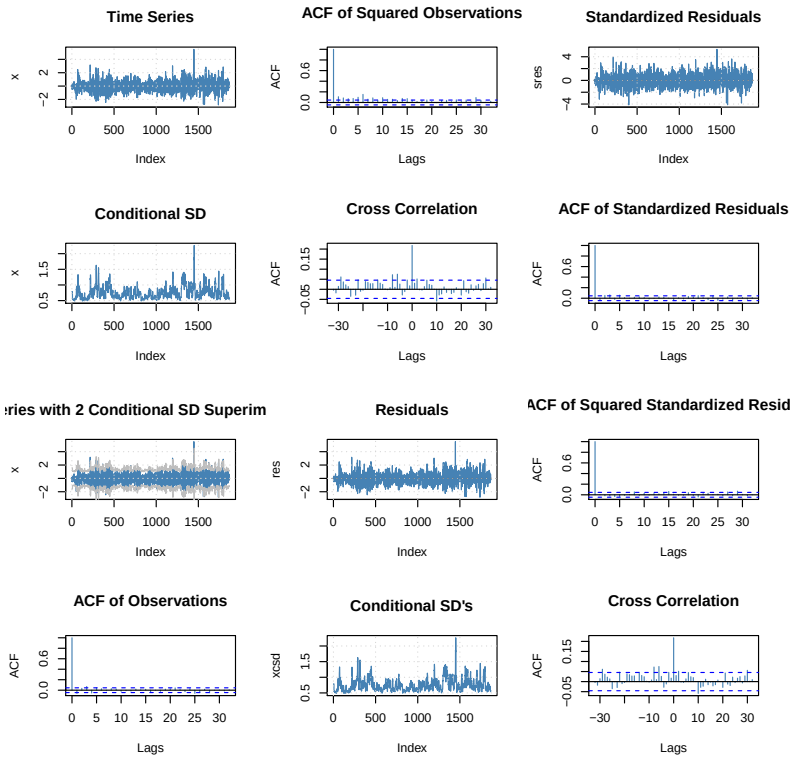

Figure B.7 fGARCH plots

```
3            0 0.5896681          0.5896681
4            0 0.6125303          0.6125303
```

The code for estimating a GARCH(1,1) model is:

```
> garch1_1 <- garchFit(formula = ~garch(1, 1), data = as.vector(et),
        trace = FALSE)
> summary(garch1_1)
Title:
 GARCH Modelling

Call:
 garchFit(formula = ~garch(1, 1), data = as.vector(et), trace = FALSE)

Mean and Variance Equation:
 data ~ garch(1, 1)
<environment: 0x000000001a033e58>
 [data = as.vector(et)]
```

```
Conditional Distribution:
 norm

Coefficient(s):
        mu       omega      alpha1       beta1
3.9861e-16  1.6309e-02  1.0937e-01  8.6873e-01

Std. Errors:
 based on Hessian

Error Analysis:
        Estimate  Std. Error  t value Pr(>|t|)
mu     3.986e-16   1.553e-02    0.000 1.000000
omega  1.631e-02   4.917e-03    3.317 0.000911 ***
alpha1 1.094e-01   1.575e-02    6.943 3.83e-12 ***
beta1  8.687e-01   1.833e-02   47.403  < 2e-16 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Log Likelihood:
 -2068.836    normalized:  -1.108701

Description:
 Fri May 24 17:26:51 2013 by user: gabriele.cantaluppi


Standardised Residuals Tests:
                                Statistic p-Value
 Jarque-Bera Test   R    Chi^2  104.7944  0
 Shapiro-Wilk Test  R    W      0.9919774 1.380026e-08
 Ljung-Box Test     R    Q(10)  27.13315  0.002480944
 Ljung-Box Test     R    Q(15)  28.75592  0.01728872
 Ljung-Box Test     R    Q(20)  37.05722  0.01151853
 Ljung-Box Test     R^2  Q(10)  4.624341  0.9148201
 Ljung-Box Test     R^2  Q(15)  8.074474  0.9207622
 Ljung-Box Test     R^2  Q(20)  15.98889  0.7173134
 LM Arch Test       R    TR^2   4.92549   0.9604214

Information Criterion Statistics:
     AIC      BIC      SIC     HQIC
2.221689 2.233547 2.221680 2.226058
```

The code for estimating an EGARCH(1,1) model, by constructing the likelihood function, follows.

```
> "Exponential GARCH likelihood"
```

```
[1] "Exponential GARCH likelihood"
> T <- length(et)
> egarchllik <- function(omega, beta, gamma, alpha) {
        sigma2 <- rep(var(et), T)
        for (i in 2:T) {
            sigma2[i] <- exp(omega + beta * log(sigma2[i -
                1]) + gamma * et[i - 1]/sigma2[i - 1]^0.5 +
                alpha * abs(et[i - 1])/sigma2[i - 1]^0.5)
        }
        llik <- -sum(dnorm(et, mean = mean(et), sd = sigma2^0.5,
            log = TRUE))
        llik
    }
> theta.start <- list(omega = 0.5, beta = 0.5, gamma = 0.5,
        alpha = 0.5)
> library(bbmle)
> out <- mle2(egarchllik, start = theta.start)
> summary(out)
Maximum likelihood estimation

Call:
mle2(minuslogl = egarchllik, start = theta.start)

Coefficients:
       Estimate Std. Error z value Pr(z)
omega -0.184652         NA      NA    NA
beta   0.966296         NA      NA    NA
gamma -0.012201         NA      NA    NA
alpha  0.213401         NA      NA    NA

-2 log L: 4133.089
> omegahat <- out@coef["omega"]
> betahat <- out@coef["beta"]
> gammahat <- out@coef["gamma"]
> alphahat <- out@coef["alpha"]
> sigma2hat <- rep(var(et), T)
> for (i in 2:T) sigma2hat[i] <- exp(omegahat + betahat *
        log(sigma2hat[i - 1]) + gammahat * et[i - 1]/sigma2hat[i -
        1]^0.5 + alphahat * abs(et[i - 1])/sigma2hat[i -
        1]^0.5)
```

Finally the conditional standard deviations implied by the preceding models can be represented by using the function `xyplot`, see Fig. B.8.

```
> xyplot(as.ts(cbind(arch6 = arch6@sigma.t, garch1_1 = garch1_1@sigma.t,
        egarch = sigma2hat^0.5)[1767:1866, ]), superpose = TRUE)
```

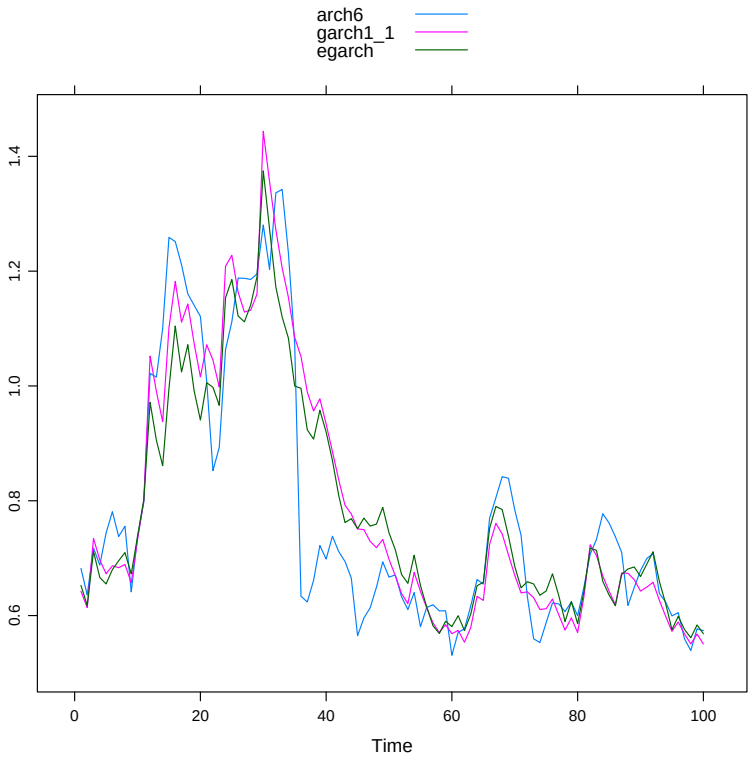

Figure B.8 Conditional standard deviations implied by the preceding models

B.4 Long-run Purchasing Power Parity (Part 1) (Section 8.5 third edition)

To import the data from the file `PPP.WF1`, which is a work file of EViews, use first the package `hexView` and next the command `readEViews`.

```
> library(hexView)
> ppp <- readEViews(unzip("purchasing_power_parity.zip",
        "PPP.WF1"))
Skipping boilerplate variable
Skipping boilerplate variable
```

Observations from January 1981 to June 1996 (T=186) on price indices and exhange rates for France and Italy (Source: datastream) are present in the file.

- `lnit` log price index Italy
- `lnfr` log price index France
- `lnp` lnit-lnfr

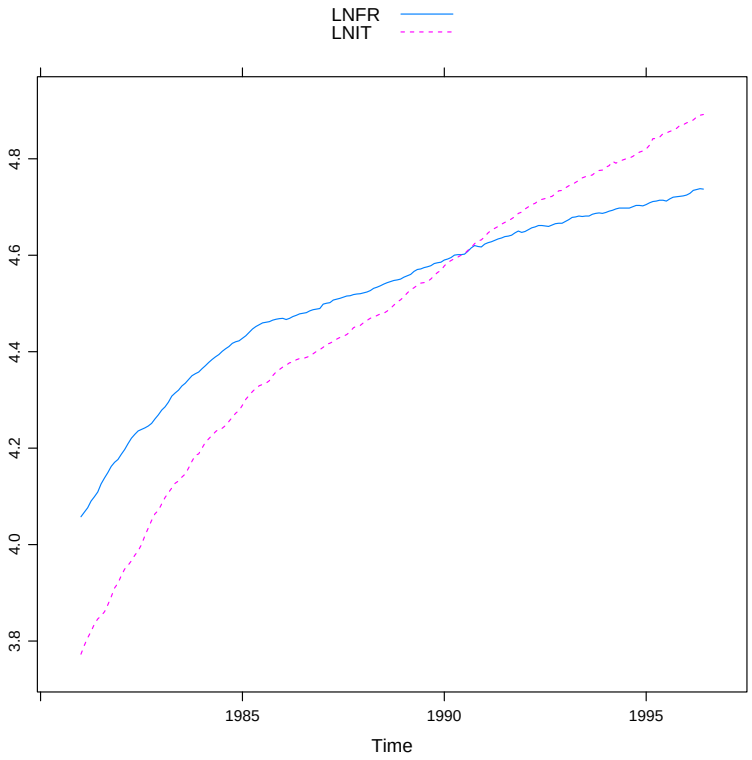

Figure B.9 Log consumer price index France and Italy, 1981:1–1996:6

- `lnx` log exchange rate France/Italy
- `cpiit` consumer price index Italy
- `cpifr` consumer price index France

By using the function `ts(object, start, frequency)` it is possible to create a multiple time series from the columns of a table; here we have to specify `freq=12` since data have a monthly frequency.

```
> ppp <- ts(ppp, start = c(1981, 1), freq = 12)


> library(lattice)
> xyplot(ppp[, 3:4], lty = c(1, 2), superpose = TRUE)
```

The Dickey-Fuller test statistic can be obtained for the French series, in the versions with only the drift and with the trend and the drift by using the following code.

```
> library(urca)
> y <- ppp[, 3]
> summary(ur.df(y, type = "drift", lags = 0))
```

```
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression drift


Call:
lm(formula = z.diff ~ z.lag.1 + 1)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0068216 -0.0014377 -0.0001496  0.0014376  0.0072683

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.0694329  0.0041997   16.53   <2e-16 ***
z.lag.1     -0.0145684  0.0009297  -15.67   <2e-16 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.002198 on 183 degrees of freedom
Multiple R-squared:  0.573,        Adjusted R-squared:  0.5706
F-statistic: 245.5 on 1 and 183 DF,  p-value: < 2.2e-16


Value of test-statistic is: -15.67 381.0699

Critical values for test statistics:
      1pct  5pct 10pct
tau2 -3.46 -2.88 -2.57
phi1  6.52  4.63  3.81
> summary(ur.df(y, type = "trend", lags = 0))
###############################################
# Augmented Dickey-Fuller Test Unit Root Test #
###############################################

Test regression trend


Call:
lm(formula = z.diff ~ z.lag.1 + 1 + tt)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0058841 -0.0013190  0.0000506  0.0012646  0.0057815
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.311e-01  1.306e-02  10.036  < 2e-16 ***
z.lag.1     -2.926e-02  3.092e-03  -9.462  < 2e-16 ***
tt           4.983e-05  1.006e-05   4.953 1.66e-06 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.002069 on 182 degrees of freedom
Multiple R-squared:  0.6237,        Adjusted R-squared:  0.6196
F-statistic: 150.8 on 2 and 182 DF,  p-value: < 2.2e-16


Value of test-statistic is: -9.4622 294.8886 150.8263

Critical values for test statistics:
      1pct  5pct 10pct
tau3 -3.99 -3.43 -3.13
phi2  6.22  4.75  4.07
phi3  8.43  6.49  5.47
```

From the preceding outputs the 5% critical values may be recovered: -2.88 for the test without the trend and -3.43 for the test with the drift and the trend.
It is possible to create a function to simultaneously run a range of Augmented Dickey-Fuller tests and reproduce Verbeek's Table 8.2.
By finding results for France and Italy separately we also consider a function `f1` which returns 1 if the corresponding (Augmented) Dickey-Fuller Test concludes for accepting the presence of a 1 root:

```
> f <- function(x) {
        nt <- summary(ur.df(y, type = "drift", lags = x))@teststat[1]
        wt <- summary(ur.df(y, type = "trend", lags = x))@teststat[1]
        return(c(nt, wt))
    }
> f1 <- function(x) {
        nt <- summary(ur.df(y, type = "drift", lags = x))
        nt <- (nt@teststat[1] >= nt@cval[1, 2])
        wt <- summary(ur.df(y, type = "trend", lags = x))
        wt <- (wt@teststat[1] >= wt@cval[1, 2])
        return(c(nt, wt))
    }
```

For France:

```
> y <- ppp[, 3]
> a <- 0:12
> names(a) <- c("DF", paste("ADF(", a[-1], ")", sep = ""))
```

```
> out <- sapply(a, f)
> out1 <- sapply(a, f1)
> rownames(out) <- rownames(out1) <- c("Without trend",
        "With trend")
> print(round(out, 3))
                   DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend -15.670 -7.148 -7.003 -4.963 -5.118 -4.114 -3.019
With trend     -9.462 -6.272 -6.933 -5.294 -6.077 -5.382 -3.919
              ADF(7) ADF(8) ADF(9) ADF(10) ADF(11) ADF(12)
Without trend -3.183 -2.878 -2.688  -2.655  -2.408  -1.763
With trend    -4.146 -3.728 -3.451  -3.592  -3.679  -2.908
> print(out1)
                  DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
With trend    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
              ADF(7) ADF(8) ADF(9) ADF(10) ADF(11) ADF(12)
Without trend  FALSE   TRUE   TRUE    TRUE    TRUE    TRUE
With trend     FALSE  FALSE  FALSE   FALSE   FALSE    TRUE
```

For Italy:

```
> y <- ppp[, 4]
> out <- sapply(a, f)
> out1 <- sapply(a, f1)
> rownames(out) <- rownames(out1) <- c("Without trend",
        "With trend")
> print(round(out, 3))
                   DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend -13.160 -6.378 -5.479 -4.407 -3.880 -3.693 -3.771
With trend     -8.403 -5.389 -5.131 -4.644 -4.289 -4.580 -5.474
              ADF(7) ADF(8) ADF(9) ADF(10) ADF(11) ADF(12)
Without trend -3.259 -2.344 -2.039  -2.113  -1.687  -0.866
With trend    -5.525 -4.529 -4.064  -3.742  -3.797  -2.997
> print(out1)
                  DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
With trend    FALSE  FALSE  FALSE  FALSE  FALSE  FALSE  FALSE
              ADF(7) ADF(8) ADF(9) ADF(10) ADF(11) ADF(12)
Without trend  FALSE   TRUE   TRUE    TRUE    TRUE    TRUE
With trend     FALSE  FALSE  FALSE   FALSE   FALSE    TRUE
```

With reference to the log exchange rate Italy-France, see Fig. B.10 obtained with the code `xyplot(ppp[,6]-ppp[,5])`, we have the following unit root tests.

```
> a <- 0:6
> names(a) <- c("DF", paste("ADF(", a[-1], ")", sep = ""))
> y <- ppp[, 6]
```

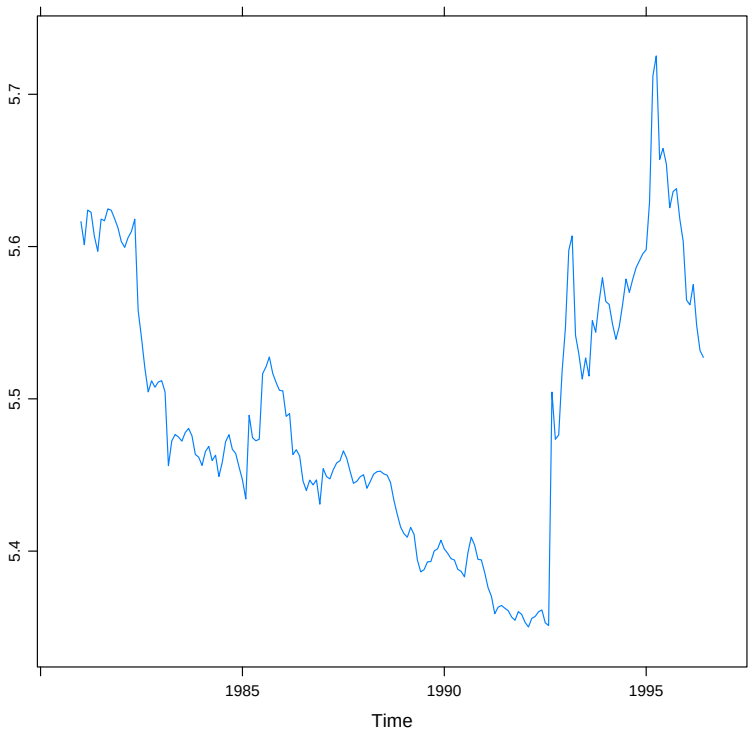

Figure B.10 Log real exchange rate Italy-France, 1981:1–1996:6

```
> out <- sapply(a, f)
> out1 <- sapply(a, f1)
> rownames(out) <- rownames(out1) <- c("Without trend",
        "With trend")
> print(round(out, 3))
                  DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend -0.328 -0.361 -0.160 -0.292 -0.366 -0.463 -0.643
With trend    -1.900 -1.884 -1.925 -2.012 -2.026 -2.032 -2.262
> print(out1)
                DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend TRUE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
With trend    TRUE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
```

While the tests for the log real exchange rate Italy-France are:

```
> y <- ppp[, 6] - ppp[, 5]
> out <- sapply(a, f)
> out1 <- sapply(a, f1)
```

```
> rownames(out) <- rownames(out1) <- c("Without trend",
        "With trend")
> print(round(out, 3))
                  DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend -1.930 -1.874 -1.930 -1.987 -1.942 -1.966 -2.286
With trend    -1.942 -1.892 -1.961 -2.022 -1.981 -2.005 -2.326
> print(out1)
                DF ADF(1) ADF(2) ADF(3) ADF(4) ADF(5) ADF(6)
Without trend TRUE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
With trend    TRUE   TRUE   TRUE   TRUE   TRUE   TRUE   TRUE
```

B.5 Long-run Purchasing Power Parity (Part 2) (Section 9.3)

The analyisis of Long-run Purchasing Power Parity, started in Section B.4 (Verbeek's Section 8.5), is continued. To import the data from the file `PPP.WF1`, which is a work file of EViews, call first the package `hexView` and next the command `readEViews`.

```
> library(hexView)
> ppp <- readEViews(unzip("purchasing_power_parity.zip",
        "PPP.WF1"))
Skipping boilerplate variable
Skipping boilerplate variable
```

Verbeek observes that the relationship $s_t = p_t - p_t^*$, where s_t, p_t and p_t^* are respectively the log of the spot exchange rate, the log of domestic prices and that of foreign prices, may be interpreted as an equilibrium (long-run) or cointegrating relationship.
In the example, observations for France and Italy from January 1981 until June 1996 are considered.
See Section B.4 for the analysis to detect the non-stationarity of the real exchange rate $rs_t = s_t - p_t + p_t^*$.
Verbeek suggests to test if the cointegrating relationship involving the log exchange rate s_t and the log of price ratio $p_t - p_t^*$ can be established. The results in Section B.4 suggested s_t as $I(1)$.
Augmented Dickey Fuller tests can be performed to establish if also the ratio $p_t - p_t^*$ is $I(1)$, by having recourse to the function `ur.df` available in the package `urca`. So Verbeek's Table 9.4 can be reproduced with the following code.

```
> library(urca)
> x <- ppp$LNP
> a <- 0:6
> names(a) <- c("DF", paste("ADF(", a[-1], ")", sep = ""))
> f <- function(k) {
        nt <- summary(ur.df(x, type = "drift", lags = k))@teststat[1]
        wt <- summary(ur.df(x, type = "trend", lags = k))@teststat[1]
        return(c(nt, wt))
```

```
    }
> out <- sapply(a, f)
> rownames(out) <- c("Without trend", "With trend")
> print(t(round(out, 3)))
       Without trend With trend
DF            -1.563     -2.692
ADF(1)        -0.993     -2.960
ADF(2)        -1.003     -2.678
ADF(3)        -1.058     -3.130
ADF(4)        -1.014     -2.562
ADF(5)        -1.294     -2.493
ADF(6)        -2.014     -3.096
```

Remembering that the 5% critical values for the Dickey Fuller statistics are -2.88, -3.43 respectively for the situations with only a drift and with both a drift and the trend, the hypothesis of non-stationarity cannot be rejected.
The parameters in the cointegrating regression (see Verbeek's Table 9.5)

$$s_t = \alpha + \beta(p_t - p_t^*) + \varepsilon_t$$

can be estimated by having, as usual, recourse to the function `lm`.

```
> a <- lm(LNX ~ LNP, data = ppp)
> summary(a)

Call:
lm(formula = LNX ~ LNP, data = ppp)

Residuals:
     Min       1Q   Median       3Q      Max
-0.13617 -0.05722 -0.01825  0.06522  0.24031

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.487197   0.006777  809.71   <2e-16 ***
LNP         0.982213   0.051328   19.14   <2e-16 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.08603 on 184 degrees of freedom
Multiple R-squared:  0.6656,        Adjusted R-squared:  0.6638
F-statistic: 366.2 on 1 and 184 DF,  p-value: < 2.2e-16
```

Tests on the residuals for establishing the possible presence of nonstationarity can also be performed.
The cointegrating regression Durbin-Watson (CRDW) statistic can be obtained with the function `dwtest` available in the package `lmtest`; pay attention to consider only the value of the statistic and not its p-value, which is computed against the null of

no autocorrelation presence and not for no cointegration. See Verbeek's Table 9.3 for the 5% critical values for the CRDW test for no cointegration.

```
> library(lmtest)
> dwtest(a)$stat
        DW
0.05546948
```

The Augmented Dickey Fuller cointegration test is also performed on the residuals, see Verbeek's Table 9.6. We can have recourse again to the function `f` constructed above, by considering only the first row of the resulting output, which is referred to the situation considering the presence of only the drift.

```
> x <- a$residuals
> a <- 0:6
> out <- sapply(a, f)
> print(t(round(out[1, ], 3)))
     [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
[1,] -1.9 -1.848 -1.897 -1.955 -1.914 -1.941 -2.259
```

Also in this case only the value of the statistic can be considered. See Verbeek's Table 9.2 for the 1%, 5% and 10% asymptotic critical values residual-based unit root ADF test for no cointegration (with constant term). In the present case the 5% critical value is -3.34 since two variables were considered in the cointegration relationship. So the null hypothesis of a unit root cannot be rejected.
Verbeek then suggests to consider a more general cointegrating relationships between the three variables s_t, p_t and p_t^*, by estimating the parameters in the model

$$s_t = \alpha + \beta p_t + \gamma p_t^* + \varepsilon_t,$$

see Verbeek's Table 9.7, and performing the corresponding tests on the residuals as made above, see Verbeek's Table 9.8.

```
> a <- lm(LNX ~ LNIT + LNFR, data = ppp)
> summary(a)
Call:
lm(formula = LNX ~ LNIT + LNFR, data = ppp)

Residuals:
     Min       1Q   Median       3Q      Max
-0.14334 -0.04029  0.00586  0.04347  0.16673

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  12.5092     0.5170   24.19   <2e-16 ***
LNIT          3.0964     0.1598   19.37   <2e-16 ***
LNFR         -4.6291     0.2710  -17.09   <2e-16 ***
---
```

```
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.06088 on 183 degrees of freedom
Multiple R-squared:  0.8335,        Adjusted R-squared:  0.8316
F-statistic: 457.9 on 2 and 183 DF,  p-value: < 2.2e-16
> library(lmtest)
> dwtest(a)$stat
       DW
0.1524492
> x <- a$residuals
> a <- 0:6
> out <- sapply(a, f)
> print(t(round(out[1, ], 3)))
       [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]
[1,] -2.798 -3.152 -2.957 -2.865 -2.858 -2.918 -2.906
```

The 5% critical value for the residual-based unit root ADF test for no cointegration (with constant term) is, from Verbeek's Table 9.2, -3.74 and again, the null hypothesis of no cointegrating relationship presence cannot be rejected. Verbeek concludes that the sample period is just not long enough to find sufficient evidence for a cointegrating relationship.

B.6 Long-run Purchasing Power Parity (Part 3) (Section 9.5.4)

The same data set of the preceding Section is considered.
The Johansen's technique is applied to analyse the existence of one or more cointegrating relationships between the three variables s_t, p_t and p_t^*.
Verbeek suggests to temporary consider $p = 3$ as the maximum order of the lags in the autoregressive representation he gives in relationship (9.42), that is:

$$\vec{Y}_t = \delta + \Theta_1 \vec{Y}_{t-1} + \Theta_2 \vec{Y}_{t-2} + \Theta_3 \vec{Y}_{t-3} + \vec{\varepsilon}_t$$

and to exclude the presence of a time trend. He remarks that the first step of the Johansen's technique consists in determining the order p.
The maximum eigenvalue tests for cointegration can be obtained by means of the function `ca.jo` available in the package `urca`, see Verbeek's Table 9.10[4].

```
> sppstar <- ppp[, c("LNX", "LNIT", "LNFR")]
> table9.10 <- ca.jo(sppstar, ecdet = "const", type = "eigen",
        K = 3, spec = "longrun")
> summary(table9.10)
```

[4] Critical values for the max-eigenvalue test are taken from Osterwald-Lenum (1992). Though quite similar, they differ somewhat from the critical values reported by Verbeek in Table 9.9. Observe that tests are reversed with respect to Verbeek's output.

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

Test type: maximal eigenvalue statistic (lambda max),
without linear trend and constant in cointegration

Eigenvalues (lambda):
[1] 3.009072e-01 1.134300e-01 3.421711e-02 1.919659e-16

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  6.37  7.52  9.24 12.97
r <= 1 | 22.03 13.75 15.67 20.20
r = 0  | 65.51 19.77 22.00 26.81

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

             LNX.l3    LNIT.l3    LNFR.l3   constant
LNX.l3     1.000000   1.000000   1.000000   1.000000
LNIT.l3   -8.195759  -5.260519  -2.987574   6.075268
LNFR.l3   15.861439   8.446387   4.246577 -10.206302
constant -41.171878 -19.917470 -11.280597  13.604985

Weights W:
(This is the loading matrix)

             LNX.l3      LNIT.l3       LNFR.l3      constant
LNX.d  -0.008012930  0.006615146 -0.0525608513 -3.388914e-14
LNIT.d -0.004476013  0.005729273  0.0007730543  2.394306e-16
LNFR.d -0.005098807 -0.002077217 -0.0004177991 -7.141077e-15
```

Johansen's tests seem to indicate the presence of two cointegrating relationships. The result is in contrast with the one obtained in the preceding Section. Verbeek suggests as a possible explanation that the number of lags, which were included, is too small, so shows what happens by including a lag of $p = 12$, since monthly data are available, see Verbeek's Table 9.11. The `ca.jo` function is again used with `K`=12.

```
> table9.11 <- ca.jo(sppstar, ecdet = "const", type = "eigen",
        K = 12, spec = "longrun")
> summary(table9.11)
######################
# Johansen-Procedure #
######################
```

```
Test type: maximal eigenvalue statistic (lambda max),
without linear trend and constant in cointegration

Eigenvalues (lambda):
[1]  1.061252e-01  9.014319e-02  3.489379e-02 -5.457996e-17

Values of teststatistic and critical values of test:

          test 10pct  5pct  1pct
r <= 2 |  6.18  7.52  9.24 12.97
r <= 1 | 16.44 13.75 15.67 20.20
r = 0  | 19.52 19.77 22.00 26.81

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

            LNX.l12   LNIT.l12   LNFR.l12   constant
LNX.l12    1.000000   1.000000   1.000000    1.00000
LNIT.l12  -6.347233  -7.434908  -2.893806  -86.47785
LNFR.l12  14.754946  13.600320   4.211233  125.54776
constant -42.341081 -34.096936 -11.493725 -181.38924

Weights W:
(This is the loading matrix)

             LNX.l12     LNIT.l12      LNFR.l12      constant
LNX.d   0.0028223030 -0.024689887 -5.932458e-02 -1.076575e-13
LNIT.d  0.0007005145 -0.003935929  4.186972e-03 -4.968664e-15
LNFR.d -0.0002058296 -0.007082123  1.751147e-05 -1.946356e-14
```

The test that considers the rejection of the presence of 0 or 1 cointegrating vectors implies a marginal rejection of the hypothesis, thought with an evidence much weaker than before. Supposing to continue the analysis with the presence of 1 cointegrating vector, the `ca.jo` summary has also provided the estimation results for the normalized cointegrating vector. Observe that the coefficients have signs opposite to those reported by Verbeek. The function `ca.jo` 'normalizes' the eigenvector by setting to 1 its first component.

EDUCatt - Ente per il Diritto allo Studio Universitario dell'Università Cattolica
Largo Gemelli 1, 20123 Milano - tel. 02.7234.22.35 - fax 02.80.53.215
e-mail: editoriale.dsu@educatt.it (produzione); librario.dsu@educatt.it (distribuzione)
web: www.educatt.it/libri

Addendum 3rd edition