

**Faculty of Business and Law**

**School of Accounting, Finance and Economics**

**BECS2002 / Econometrics and Data Analytics**

**Lab Handbook**

**Week 17**

**Module coordinator:** Dr Camilo Calderon

**Email:** cam.calderon@dmu.ac.uk

**This version: 22/01/2024**

This handbook has been produced to provide students with specific information and guidance about their labs. The contents of this handbook are indicative, this means they can be updated or modified upon the review of the module leader (Cam).

An electronic version of this handbook (which is continuously updated) is available on the VLE system, Blackboard, which you should consult regularly as the main reference point throughout your studies.

### Indicative Contents

Video 1 - GLS: Known Form of Variance 5

Lecture 1 - Time-Series: Stationary Variables 13

Lecture 2 -Estimation with Serially Correlated Errors 25

Video 2 - Random Regressors 38

Lab 1 - Simultaneous Equations Models 46

Lab 2 Time Series: Non-stationarity 53

Lab 3 - Unit Root Tests for Stationarity 61

Lab 4 Cointegration 67

Lab 5 - VEC and VAR Models 71

Lab 6 - Estimating a VAR Model 76

Optional Lab 1 - Time-Varying Volatility and ARCH Models 83

Optional Lab 2 - Panel Data Models 91

Video 1 - GLS: Known Form of Variance

Prerequisite:

**rm**(list=**ls**()) *#Removes all items in Environment!* **library**(lmtest) *#for coeftest() and bptest().*

**library**(broom) *#for glance() and tidy()*

**library**(PoEdata) *#for PoE4 datasets*

**library**(car) *#for hccm() robust standard errors*

**library**(sandwich)

**library**(knitr)

**library**(stargazer)

Let us consider the regression equation given in Equation 8.10), where the errors are assumed heteroskedastic.

*yi* = *β*1 + *β*2*xi* + *ei, var*(*ei*) = *σi* (8.10)

Heteroskedasticity implies different variances of the error term for each observation. Ideally, one should be able to estimate the *N* variances in order to obtain reliable standard errors, but this is not possible. The second best in the absence of such estimates is an assumption of how variance depends on one or several of the regressors. The estimator obtained when using such an assumption is called a **generalized least squares** estimator, **gls**, which may involve a structure of the errors as proposed in Equation 8.11, which assumes a linear relationship between variance and the regressor *xi* with the unknown parameter *σ*2 as a proportionality factor.

Table 8.7: OLS estimates for the ’food’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 83.4160 | 43.41016 | 1.92158 | 0.062182 |
| Income | 10.2096 | 2.09326 | 4.87738 | 0.000019 |

*var*(*ei*) = *σi*2 = *σ*2*xi* (8.11)

One way to circumvent guessing a proportionality factor in Equation 8.11 is to transform the initial model in Equation 8.10 such that the error variance in the new model has the structure proposed in Equation 8.11. This can be achieved if the initial model is divided through by and estimate the new model shown in Equation 8.12. If Equation 8.12 is correct, then the resulting estimator is BLUE.

*yi∗  = β*1*x∗i1 + β*2*x∗i2 + e∗i* (8.12)

In general, if the initial variables are multiplied by quantities that are specific to each observation, the resulting estimator is called a **weighted least squares** estimator, **wls**. Unlike the robust standard errors method for heteroskedasticity correction, **gls** or **wls** methods change the estimates of regression coefficients.

The function lm() can do **wls** estimation if the argument weights is provided under the form of a vector of the same size as the other variables in the model. *R* takes the square roots of the weights provided to multiply the variables in the regression. Thus, if you wish to multiply the model by , the weights should be *wi* = .

Let us apply these ideas to re-estimate the *food* equation, which we have determined to be affected by heteroskedasticity.

w <- 1/food$income

food.wls <- **lm**(food\_exp~income, weights=w, data=food)

vcvfoodeq <- **coeftest**(foodeq, vcov.=cov1)

**kable**(**tidy**(foodeq), caption="OLS estimates for the 'food' equation")

**kable**(**tidy**(food.wls),

caption="WLS estimates for the 'food' equation" )

**kable**(**tidy**(vcvfoodeq),caption= "OLS estimates for the 'food' equation with robust standard errors" )

Tables 8.7, 8.8, and 8.9 compare ordinary least square model to a weighted least squares model and to OLS with robust standard errors.

Table 8.8: WLS estimates for the ’food’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 78.6841 | 23.78872 | 3.30762 | 0.002064 |
| income | 10.4510 | 1.38589 | 7.54100 | 0.000000 |

Table 8.9: OLS estimates for the ’food’ equation with robust standard errors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 83.4160 | 27.46375 | 3.03731 | 0.004299 |
| income | 10.2096 | 1.80908 | 5.64356 | 0.000002 |

*√*

The WLS model multiplies the variables by , where the weights provided have to be *w* = 1*/income*. The effect of introducing the weights is a slightly lower intercept and, more importantly, different standard errors. Please note that the WLS standard errors are closer to the robust (HC1) standard errors than to the OLS ones.

## 8.5 Grouped Data

We have seen already (Equation 8.7) how a dichotomous indicator variable splits the data in two groups that may have different variances. The generalized least squares method can account for group heteroskedasticity, by choosing appropriate weights for each group; if the variables are transformed by multiplying them by 1*/σj*, for group *j*, the resulting model is homoskedastic. Since *σj* is unknown, we replace it with its estimate *σ*ˆ*j*. This method is named **feasible generalized least squares**.

|  |
| --- |
| **data**("cps2", package="PoEdata")  rural.lm <- **lm**(wage~educ+exper, data=cps2, subset=(metro==0))  sigR <- **summary**(rural.lm)$sigma  metro.lm <- **lm**(wage~educ+exper, data=cps2, subset=(metro==1))  sigM <- **summary**(metro.lm)$sigma  cps2$wght <- **rep**(0, **nrow**(cps2))  *# Create a vector of weights*  for (i in 1:1000)  { if (cps2$metro[i]==0){cps2$wght[i] <- 1/sigR^2} else{cps2$wght[i] <- 1/sigM^2}  }  wge.fgls <- **lm**(wage~educ+exper+metro, weights=wght, data=cps2)  wge.lm <- **lm**(wage~educ+exper+metro, data=cps2) |
| wge.hce <- **coeftest**(wge.lm, vcov.=**hccm**(wge.lm, data=cps2))  **stargazer**(rural.lm, metro.lm, wge.fgls,wge.hce,  header=FALSE,  title="OLS vs. FGLS estimates for the 'cps2' data",  type=.stargazertype, *# "html" or "latex" (in index.Rmd)*  keep.stat="n", *# what statistics to print*  omit.table.layout="n",  star.cutoffs=NA,  digits=3,  *# single.row=TRUE,*  intercept.bottom=FALSE, *#moves the intercept coef to top*  column.labels=**c**("Rural","Metro","FGLS", "HC1"),  dep.var.labels.include = FALSE,  model.numbers = FALSE,  dep.var.caption="Dependent variable: wage",  model.names=FALSE,  star.char=NULL) *#supresses the stars* |

Table 8.10: OLS vs. FGLS estimates for the ’cps2’ data

Dependent variable: wage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Rural | Metro | FGLS | HC1 |
| Constant | *−*6.166 | *−*9.052 | *−*9.398 | *−*9.914 |
|  | (1.899) | (1.189) | (1.020) | (1.218) |
| educ | 0.956 | 1.282 | 1.196 | 1.234 |
|  | (0.133) | (0.080) | (0.069) | (0.084) |
| Exper | 0.126 | 0.135 | 0.132 | 0.133 |
|  | (0.025) | (0.018) | (0.015) | (0.016) |
| Metro |  |  | 1.539 | 1.524 |
|  |  |  | (0.346) | (0.346) |
| Observations | 192 | 808 | 1,000 |  |

The table titled “OLS, vs. FGLS estimates for the ‘cps2’ data” helps comparing the coefficients and standard errors of four models: OLS for rural area, OLS for metro area, feasible GLS with the whole dataset but with two types of weights, one for each area, and, finally, OLS with heteroskedasticity-consistent (HC1) standard errors. Please be reminded that the regular OLS standard errors are not to be trusted in the presence of heteroskedasticity.

The previous code sequence needs some explanation. It runs two regression models, rural.lm and metro.lm just to estimate *σ*ˆ*R* and *σ*ˆ*M* needed to calculate the weights for each group. The subsets, this time, were selected directly in the lm() function through the argument subset=, which takes as argument some logical expression that may involve one or more variables in the dataset. Then, I create a new vector of a size equal to the number of observations in the dataset, a vector that will be populated over the next few code lines with weights. I choose to create this vector as a new column of the dataset cps2, a column named wght. With this the hard part is done; I just need to run an lm() model with the option weights=wght and that gives my FGLS coefficients and standard errors.

The next lines make a for loop runing through each observation. If observation *i* is a rural area observation, it receives a weight equal to 1*/σR*2 ; otherwise, it receives the weight 1*/σM*2 . Why did I square those *sigmas*? Because, remember, the argument weights in the lm() function requires the square of the factor multiplying the regression model in the WLS method.

The remaining part of the code repeats models we ran before and places them in one table for making comparison easier.

## 8.6 GLS: Unknown Form of Variance

Suppose we wish to estimate the model in Equation 8.13, where the errors are known to be heteroskedastic but their variance is an unknown function of *S* some variables *zs* that could be among the regressors in our model or other variables.

*yi* = *β*1 + *β*2*xi*2 + *...βkxiK* + *ei* (8.13)

Equation 8.14 uses the residuals from Equation 8.13 as estimates of the variances of the error terms and serves at estimating the functional form of the variance. If the assumed functional form of the variance is the exponential function *var*(*ei*) = *σi*2 = *σ*2*xγi* , then the regressors *zis* in Equation 8.14 are the logs of the initial regressors *xis*, *zis* = *log*(*xis*).

*ln*(*e*ˆ2*i* ) = *α*1 + *α*2*zi*2 + *...* + *αSziS* + *νi* (8.14)

The variance estimates for each error term in Equation 8.13 are the fitted values, *σ*ˆ*i*2 of Equation 8.14, which can then be used to construct a vector of weights for the regression model in Equation 8.13. Let us follow these steps on the *food* basic equation where we assume that the variance of error term *i* is an unknown exponential function of income. So, the purpose of the following code fragment is to determine the weights and to supply them to the lm() function. Remember, lm() **multiplies** each observation by the **square root** of the weight you supply. For instance, if you want to multiply the observations by 1*/σi*, you should supply the weight *wi* = 1*/σi*2.

|  |
| --- |
| **data**("food", package="PoEdata")  food.ols <- **lm**(food\_exp~income, data=food)  ehatsq <- **resid**(food.ols)^2  sighatsq.ols <- **lm**(**log**(ehatsq)~**log**(income), data=food)  vari <- **exp**(**fitted**(sighatsq.ols))  food.fgls <- **lm**(food\_exp~income, weights=1/vari, data=food) |
| **stargazer**(food.ols, food.HC1, food.wls, food.fgls,  header=FALSE,  title="Comparing various 'food' models",  type=.stargazertype, *# "html" or "latex" (in index.Rmd)*  keep.stat="n", *# what statistics to print*  omit.table.layout="n",  star.cutoffs=NA,  digits=3,  *# single.row=TRUE,*  intercept.bottom=FALSE, *#moves the intercept coef to top*  column.labels=**c**("OLS","HC1","WLS","FGLS"),  dep.var.labels.include = FALSE,  model.numbers = FALSE,  dep.var.caption="Dependent variable: 'food expenditure'",  model.names=FALSE,  star.char=NULL) *#supresses the stars* |

Table 8.11: Comparing various ’food’ models

Dependent variable: ’food expenditure’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | OLS | HC1 | WLS | FGLS |
| Constant | 83.416 | 83.416 | 78.684 | 76.054 |
|  | (43.410) | (27.464) | (23.789) | (9.713) |
| Income | 10.210 | 10.210 | 10.451 | 10.633 |
|  | (2.093) | (1.809) | (1.386) | (0.972) |
| Observations | 40 |  | 40 | 40 |

The table titled “Comparing various ‘food’ models” shows that the FGLS with unknown variances model substantially lowers the standard errors of the coefficients, which in turn increases the *t*-ratios (since the point estimates of the coefficients remain about the same), making an important difference for hypothesis testing.

For a few classes of variance functions, the weights in a GLS model can be calculated in *R* using the varFunc() and varWeights() functions in the package nlme.

## 8.7 Heteroskedasticity in the Linear Probability Model

As we have already seen, the linear probability model is, by definition, heteroskedastic, with the variance of the error term given by its binomial distribution parameter *p*, the probability that *y* is equal to 1, *var*(*y*) = *p*(1*−p*), where *p* is defined in Equation 8.15.

*p* = *β*1 + *β*2*x*2 + *...* + *βKxK* + *e* (8.15)

Thus, the linear probability model provides a known variance to be used with GLS, taking care that none of the estimated variances is negative. One way to avoid negative or greater than one probabilities is to artificially limit them to the interval (0*,*1).

Let us revise the *coke* model in dataset coke using this structure of the variance.

|  |
| --- |
| **data**("coke", package="PoEdata")  coke.ols <- **lm**(coke~pratio+disp\_coke+disp\_pepsi, data=coke)  coke.hc1 <- **coeftest**(coke.ols, vcov.=**hccm**(coke.ols, type="hc1"))  p <- **fitted**(coke.ols)  *# Truncate negative or >1 values of p*  pt<-p  pt[pt<0.01] <- 0.01 pt[pt>0.99] <- 0.99  sigsq <- pt\*(1-pt)  wght <- 1/sigsq  coke.gls.trunc <- **lm**(coke~pratio+disp\_coke+disp\_pepsi,  data=coke, weights=wght)  *# Eliminate negative or >1 values of p*  p1 <- p  p1[p1<0.01 | p1>0.99] <- NA  sigsq <- p1\*(1-p1)  wght <- 1/sigsq |

coke.gls.omit <- **lm**(coke~pratio+disp\_coke+disp\_pepsi, data=coke, weights=wght)

|  |
| --- |
| **stargazer**(coke.ols, coke.hc1, coke.gls.trunc, coke.gls.omit,  header=FALSE,  title="Comparing various 'coke' models",  type=.stargazertype, *# "html" or "latex" (in index.Rmd)*  keep.stat="n", *# what statistics to print*  omit.table.layout="n",  star.cutoffs=NA,  digits=4,  *# single.row=TRUE,*  intercept.bottom=FALSE, *#moves the intercept coef to top* column.labels=**c**("OLS","HC1","GLS-trunc","GLS-omit"),  dep.var.labels.include = FALSE, model.numbers = FALSE,  dep.var.caption="Dependent variable: 'choice of coke'", model.names=FALSE,  star.char=NULL) *#supresses the stars* |

Table 8.12: Comparing various ’coke’ models

Dependent variable: ’choice of coke’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | OLS | HC1 | GLS-trunc | GLS-omit |
| Constant | 0.8902 | 0.8902 | 0.6505 | 0.8795 |
|  | (0.0655) | (0.0653) | (0.0568) | (0.0594) |
| pratio | *−*0.4009 | *−*0.4009 | *−*0.1652 | *−*0.3859 |
|  | (0.0613) | (0.0604) | (0.0444) | (0.0527) |
| disp\_coke | 0.0772 | 0.0772 | 0.0940 | 0.0760 |
|  | (0.0344) | (0.0339) | (0.0399) | (0.0353) |
| disp\_pepsi | *−*0.1657 | *−*0.1657 | *−*0.1314 | *−*0.1587 |
|  | (0.0356) | (0.0344) | (0.0354) | (0.0360) |
| Observations | 1,140 |  | 1,140 | 1,124 |

Lecture 1 - Time-Series: Stationary Variables

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!*  **library**(dynlm) *#for the `dynlm()` function*  **library**(orcutt) *# for the `cochrane.orcutt()` function*  **library**(nlWaldTest) *# for the `nlWaldtest()` function*  **library**(zoo) *# for time series functions (not much used here)*  **library**(pdfetch) *# for retrieving data (just mentioned here)*  **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`*  **library**(PoEdata) *#for PoE4 datasets*  **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for kable()*  **library**(forecast) |

New packages: dynlm (Zeileis 2016); orcutt (Spada, Quartagno, and Tamburini

2012); nlWaldTest (Komashko 2016); zoo [R-zoo]; pdfetch (Reinhart 2015); and forecast (Hyndman 2016).

Time series are data on several variables on an observational unit (such as an individual, country, or firm) when observations span several periods. Correlation among subsequent observations, the importance of the natural order in the data and dynamics (past values of data influence present and future values) are features of time series that do not occur in cross-sectional data.

Time series models assume, in addition to the usual linear regression assumptions, that the series is **stationary**, that is, the distribution of the error term, as well as the correlation between error terms a few periods apart are constant over time. Constant distribution requires, in particular, that the variable does not display a trend in its mean or variance; constant correlation implies no clustering of observations in certain periods.

## 9.1 An Overview of Time Series Tools in *R*

*R* creates a time series variable or dataset using the function ts(), with the following main arguments: your data file in matrix or data frame form, the start period, the end period, the frequency of the data (1 is annual, 4 is quarterly, and 12 is monthly), and the names of your column variables. Another class of time series objects is created by the function zoo() in the package zoo, which, unlike ts(), can handle irregular or high-frequency time series. Both ts and zoo classes of objects can be used by the function dynlm() in the package with the same name to solve models that include lags and other time series specific operators.

In standard *R*, two functions are very useful when working with time series: the *difference* function, *diff*(*yt*) = *yt − yt−*1, and the *lag* function, *lag*(*yt*) = *yt−*1.

The package pdfetch is a very useful tool for getting *R*-compatible time series data from different online sources such as the World Bank, Eurostat, European Central Bank, and Yahoo Finance. The package WDI retrieves data from the very rich World Development Indicators database, maintained by the World Bank.

## 9.2 Finite Distributed Lags

A *finite distributed lag* model (FDL) assumes a linear relationship between a dependent variable *y* and several lags of an independent variable *x*. Equation 9.1 shows a finite distributed lag model **of order *q*.**

*yt* = *α* + *β*0*xt* + *β*1*xt−*1 + *...* + *βqxt−q* + *et* (9.1)

The coefficient *βs* is an *s***-period delay multiplier**, and the coefficient *β*0, the immediate (contemporaneous) impact of a change in *x* on *y*, is an **impact multiplier**. If *x* increases by one unit today, the change in *y* will be *β*0+*β*1+*...*+*βs* after *s* periods; this quantity is called the *s*-period **interim multiplier**. The **total multiplier** is equal to the sum of all *β*s in the model.

Let us look at Okun’s law as an example of an FDL model. Okun’s law relates contemporaneous (time *t*) change in unemployment rate, *DUt*, to present and past levels of economic growth rate, *Gt−s*.

Table 9.1: The ‘okun‘ dataset with differences and lags

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| g | u | uL1 | du | gL1 | gL2 | gL3 |
| 1.4 | 7.3 | NA | NA | NA | NA | NA |
| 2.0 | 7.2 | 7.3 | -0.1 | 1.4 | NA | NA |
| 1.4 | 7.0 | 7.2 | -0.2 | 2.0 | 1.4 | NA |
| 1.5 | 7.0 | 7.0 | 0.0 | 1.4 | 2.0 | 1.4 |
| 0.9 | 7.2 | 7.0 | 0.2 | 1.5 | 1.4 | 2.0 |
| 1.5 | 7.0 | 7.2 | -0.2 | 0.9 | 1.5 | 1.4 |

Table 9.1 shows how lags and differences work. Please note how each lag uses up an observation period.

Table 9.2: The ‘okun‘ distributed lag model with three lags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.5810 | 0.0539 | 10.7809 | 0.0000 |
| L(g, 0:3)0 | -0.2021 | 0.0330 | -6.1204 | 0.0000 |
| L(g, 0:3)1 | -0.1645 | 0.0358 | -4.5937 | 0.0000 |
| L(g, 0:3)2 | -0.0716 | 0.0353 | -2.0268 | 0.0456 |
| L(g, 0:3)3 | 0.0033 | 0.0363 | 0.0911 | 0.9276 |

|  |
| --- |
| **data**("okun", package="PoEdata")  **library**(dynlm)  check.ts <- **is.ts**(okun) *# "is structured as time series?"*  okun.ts <- **ts**(okun, start=**c**(1985,2), end=**c**(2009,3),frequency=4)  okun.ts.tab <- **cbind**(okun.ts, **lag**(okun.ts[,2], -1), **diff**(okun.ts[,2], lag=1),  **lag**(okun.ts[,1], -1), **lag**(okun.ts[,1], -2), **lag**(okun.ts[,1], -3))  **kable**(**head**(okun.ts.tab), caption="The `okun` dataset with differences and lags", col.names=**c**("g","u","uL1","du","gL1","gL2","gL3")) |

okunL3.dyn <- **dynlm**(**d**(u)~**L**(g, 0:3), data=okun.ts) **kable**(**tidy**(**summary**(okunL3.dyn)), digits=4,

caption="The `okun` distributed lag model with three lags")

Table 9.3: The ‘okun‘ distributed lag model with two lags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.5836 | 0.0472 | 12.3604 | 0.0000 |
| L(g, 0:2)0 | -0.2020 | 0.0324 | -6.2385 | 0.0000 |
| L(g, 0:2)1 | -0.1653 | 0.0335 | -4.9297 | 0.0000 |
| L(g, 0:2)2 | -0.0700 | 0.0331 | -2.1152 | 0.0371 |

Tables 9.2 and 9.3 summarize the results of linear models with 3 and 2 lags respectively. Many of the output analysis functions that we have used with the lm() function, such as summary() and coeftest() are also applicable to dynlm().

Table 9.4: Goodness-of-fit statistics for ‘okun‘ models

|  |  |  |  |
| --- | --- | --- | --- |
| r.squared | statistic | AIC | BIC |
| 0.652406 | 42.2306 | -55.4318 | -40.1085 |
| 0.653946 | 57.9515 | -58.9511 | -46.1293 |

Table 9.4 compares the two FDL models of the *okun* example. The first row is the model with three lags, the second is the model with two lags. All the measures in this table points to the second model (two lags) as a better specification.

okunL2.dyn <- **dynlm**(**d**(u)~**L**(g, 0:2), data=okun.ts) **kable**(**tidy**(**summary**(okunL2.dyn)), digits=4, caption="The `okun` distributed lag model with two lags")

glL3 <- **glance**(okunL3.dyn)[**c**("r.squared","statistic","AIC","BIC")] glL2 <- **glance**(okunL2.dyn)[**c**("r.squared","statistic","AIC","BIC")]

tabl <- **rbind**(glL3, **as.numeric**(glL2)) **kable**(tabl, caption="Goodness-of-fit statistics for `okun` models")

A note on how these tables were created in *R* is of interest. Table 9.1 was created using the function cbind, which puts together several columns (vectors); table 9.4 used two functions: rbind(), which puts together two rows, and as.numeric, which extracts only the numbers from the glance object, without the names of the columns.

## 9.3 Serial Correlation

**Serial correlation**, or **autocorrelation** in a time series describes the correlation between two observations separated by one or several periods. Time series tend to display autocorrelation more than cross sections because of their ordered nature.

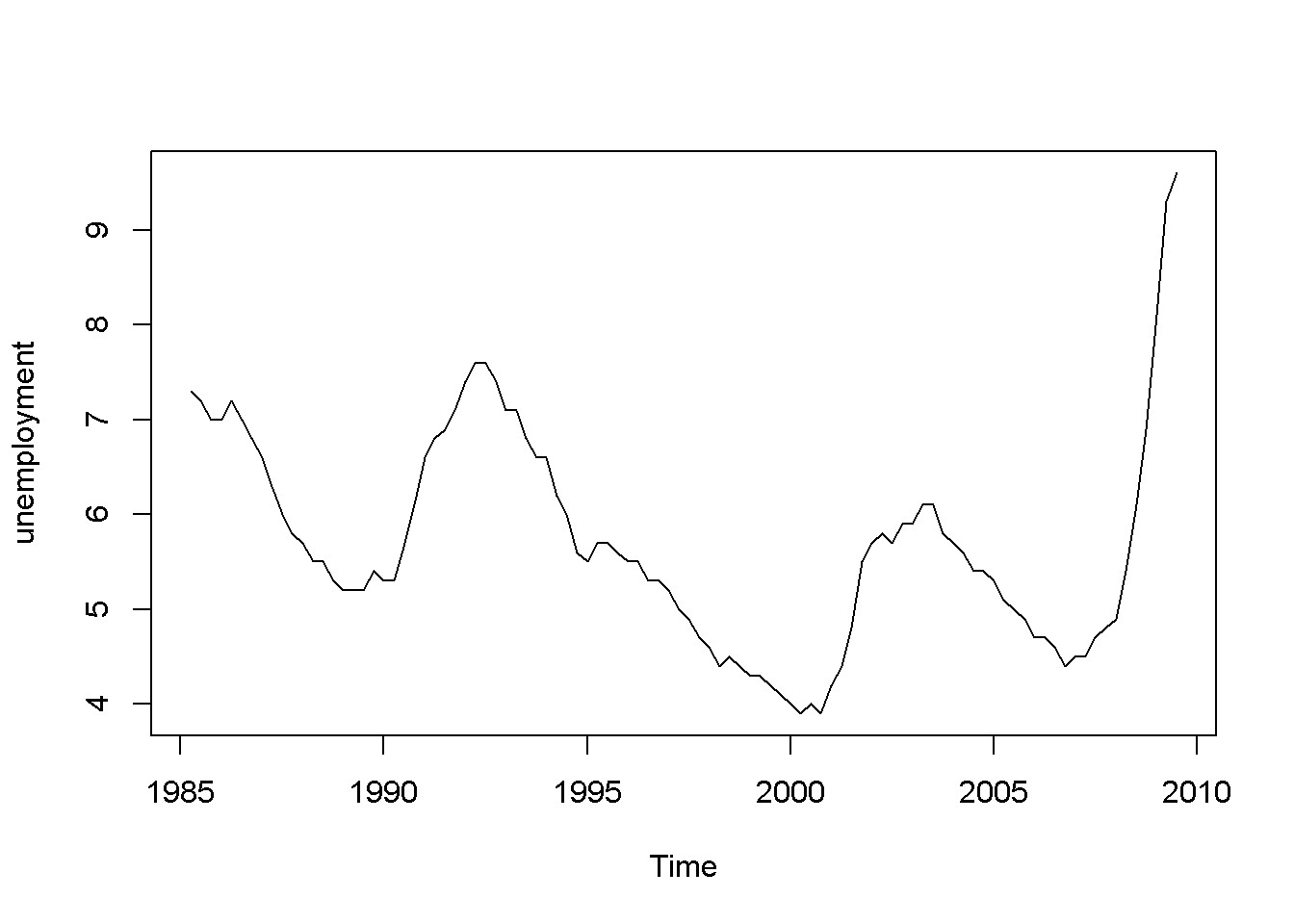
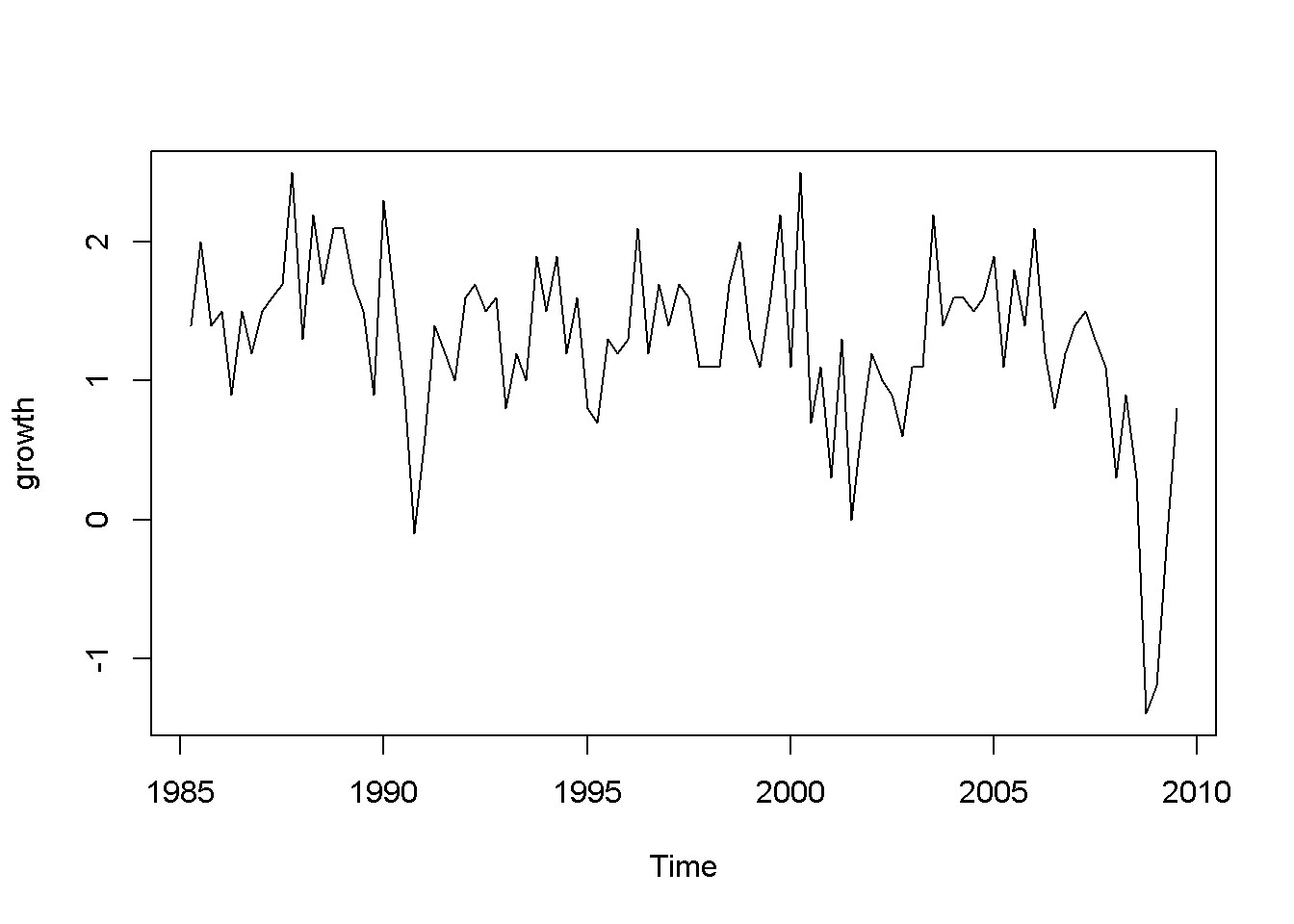


Figure 9.1: Growth and unemployment rates in the ’okun’ dataset

Autocorrelation could be an attribute of one series, independent of the model in which this series appears. If this series is, however, an error term, its properties do depend on the model, since error series can only exist in relation to a model.

**plot**(okun.ts[,"g"], ylab="growth")

**plot**(okun.ts[,"u"], ylab="unemployment")

The “growth” graph in Figure 9.1 display clusters of values: positive for several periods followed by a few of negative values, which is an indication of autocorrelation; the same is true for unemployment, which does not change as dramatically as growth but still shows persistence.

|  |
| --- |
| ggL1 <- data.frame(cbind(okun.ts[,"g"], lag(okun.ts[,"g"],-1)))  names(ggL1) <- c("g","gL1")  plot(ggL1)  meang <- mean(ggL1$g, na.rm=TRUE)  abline(v=meang, lty=2)  abline(h=mean(ggL1$gL1, na.rm=TRUE), lty=2)  ggL2 <- data.frame(cbind(okun.ts[,"g"], lag(okun.ts[,"g"],-2)))  names(ggL2) <- c("g","gL2")  plot(ggL2)  meang <- mean(ggL2$g, na.rm=TRUE)  abline(v=meang, lty=2)  abline(h=mean(ggL2$gL2, na.rm=TRUE), lty=2) |

Figures 9.2 illustrate the correlation between the growth rate and its first two lags, which is, indeed autocorrelation. But is there a more precise test to detect autocorrelation?

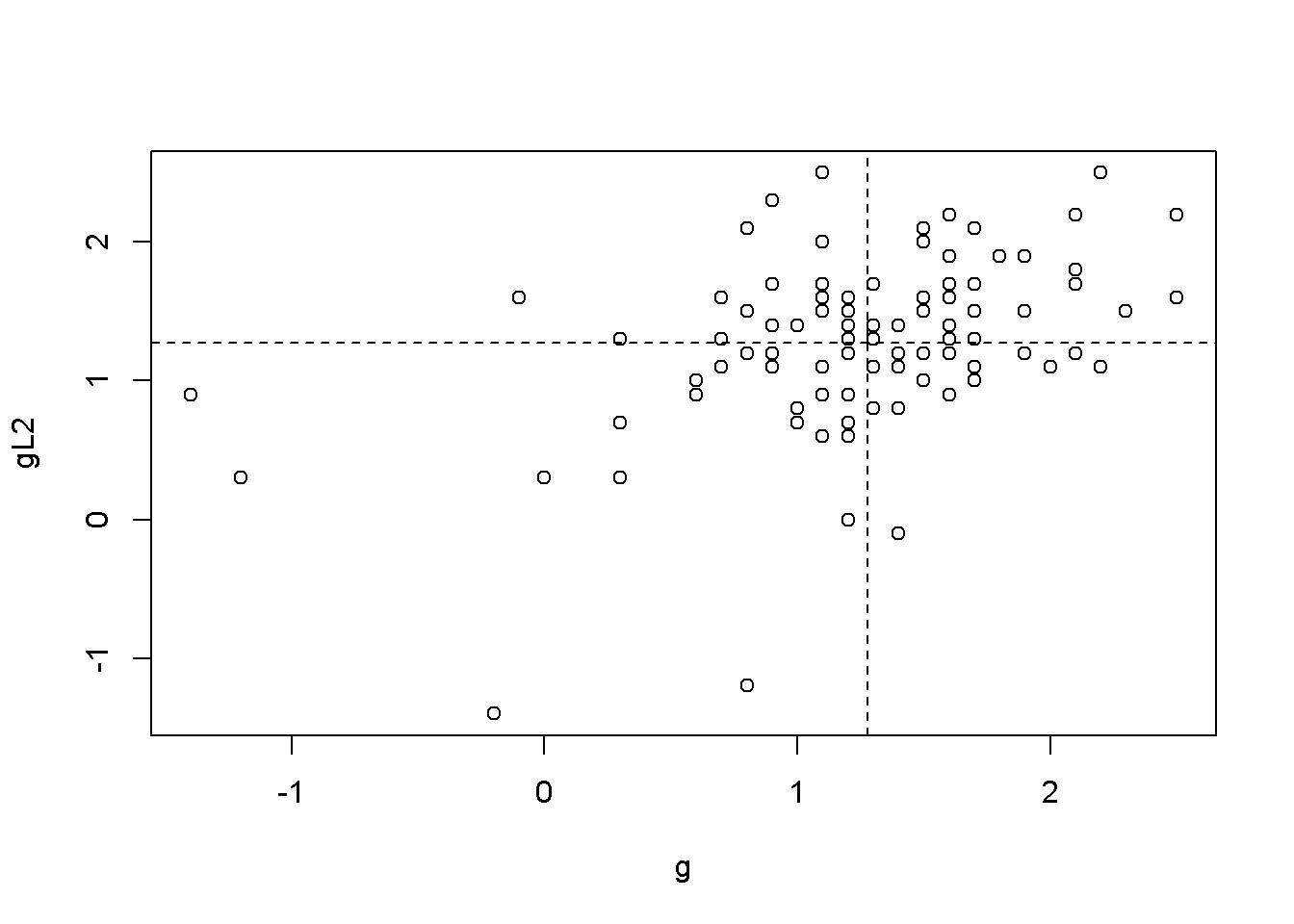
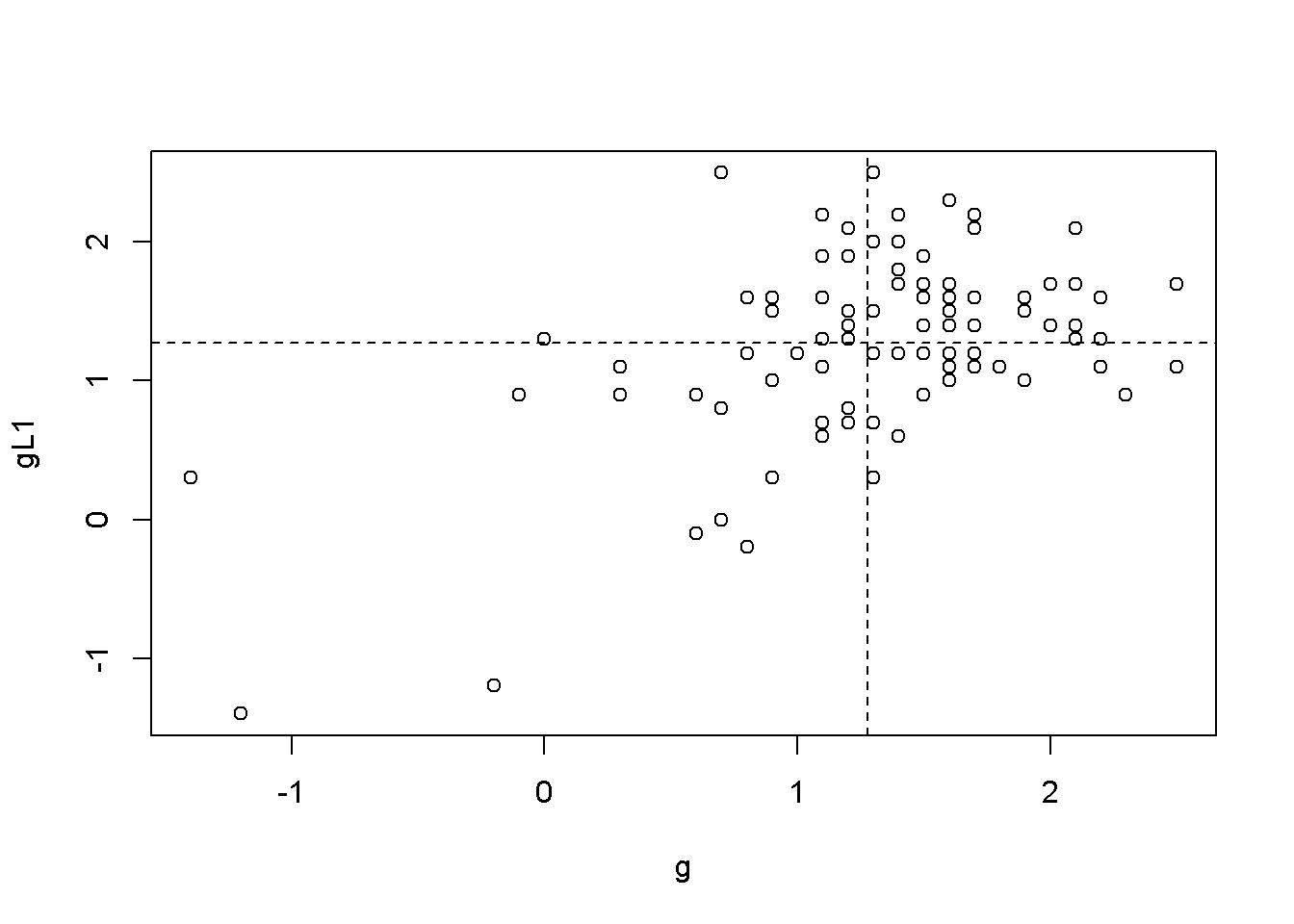


Figure 9.2: Scatter plots between ’g’ and its lags

Suppose we wish to test the hypothesis formulated in Equation 9.2, where *ρk* is the population *k*-th order autocorrelation coefficient.

A black text on a white background

Description automatically generated (9.2)

A test can be constructed based on the sample correlation coefficient, *rk*, which measures the correlation between a variable and its *k*-th lag; the test statistic is given in Equation 9.3, where *T* is the number of periods.

A math equation with black text

Description automatically generated (9.3)

For a 5% significance level, *Z* must be outside the interval [*−*1*.*96*,*1*,*96], that is, in the rejection region. Rejecting the null hypothesis is, in this case, bad news, since rejection constitutes evidence of autocorrelation. So, for a way to remember the meaning of the test, one may think of it as a **test of non-autocorrelation**.

The results of the (non-) autocorrelation test are usually summarized in a**correlogram**, a bar diagram that visualizes the values of the test statistic for several lags as well as the 95% confidence interval. A bar ( *Trk*) that exceeds (upward or downward) the limits of the confidence interval indicates autocorrelation for the corresponding lag.

growth\_rate <- okun.ts[,"g"]

**acf**(growth\_rate)

Figure 9.3 is a correlogram, where each bar corresponding to one lag, starting with lag 0. The correlogram shows little or no evidence of autocorrelation, except for the first and second lag (second and third bar in the figure).

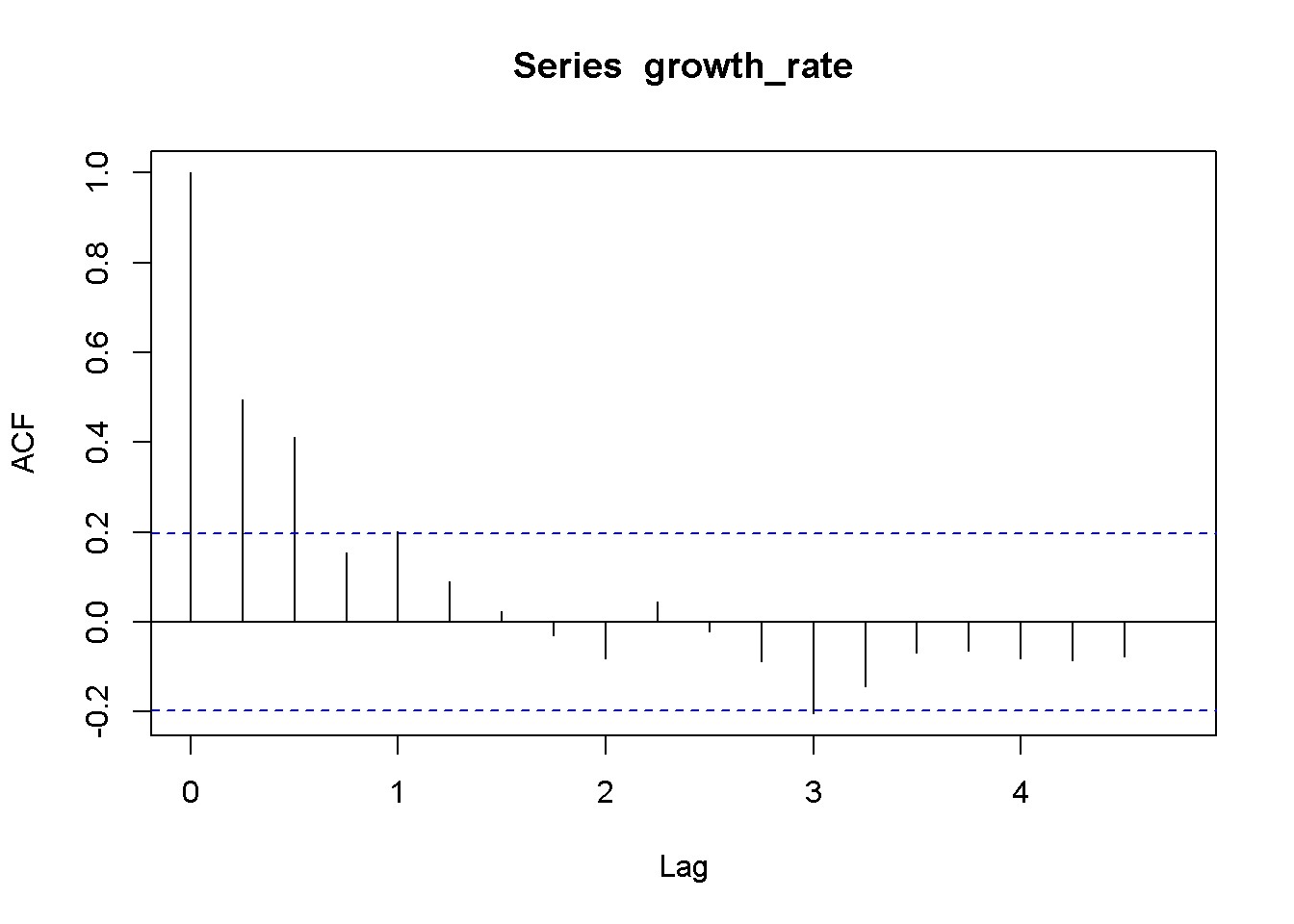


Figure 9.3: Correlogram for the growth rate, dataset ’okun’

Let us consider another example, the dataset *phillips\_aus*, which containes quarterly data on unemploymnt and inflation over the period 1987Q1 to 2009Q3. We wish to apply the autocorrelation test to the error term in a time series regression to see if the non-autocorrelation in the errors is violated. Let us consider the FDL model in Equation 9.4.

*inft* = *β*1 + *β*2*Dut* + *et* (9.4)

Let’s first take a look at plots of the data (visualising the data is said to be a good practice rule in data analysis.)

|  |
| --- |
| **data**("phillips\_aus", package="PoEdata")  phill.ts <- **ts**(phillips\_aus,  start=**c**(1987,1),  end=**c**(2009,3),  frequency=4)  inflation <- phill.ts[,"inf"]  Du <- **diff**(phill.ts[,"u"])  **plot**(inflation)  **plot**(Du) |

The plots in Figure 9.4 definitely show patterns in the data for both inflation and unemployment rates. But we are interested to determine if the error term in Equation

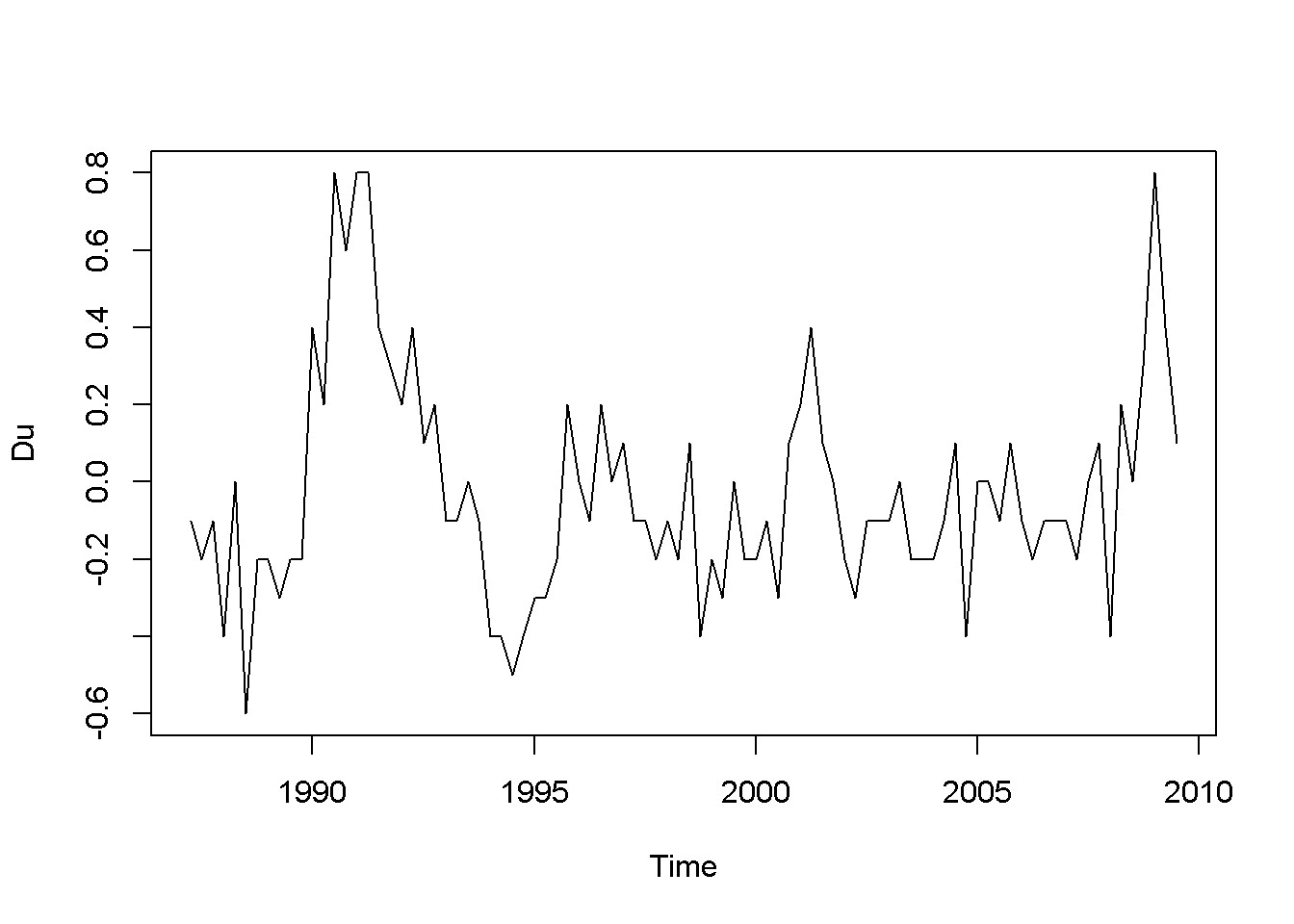
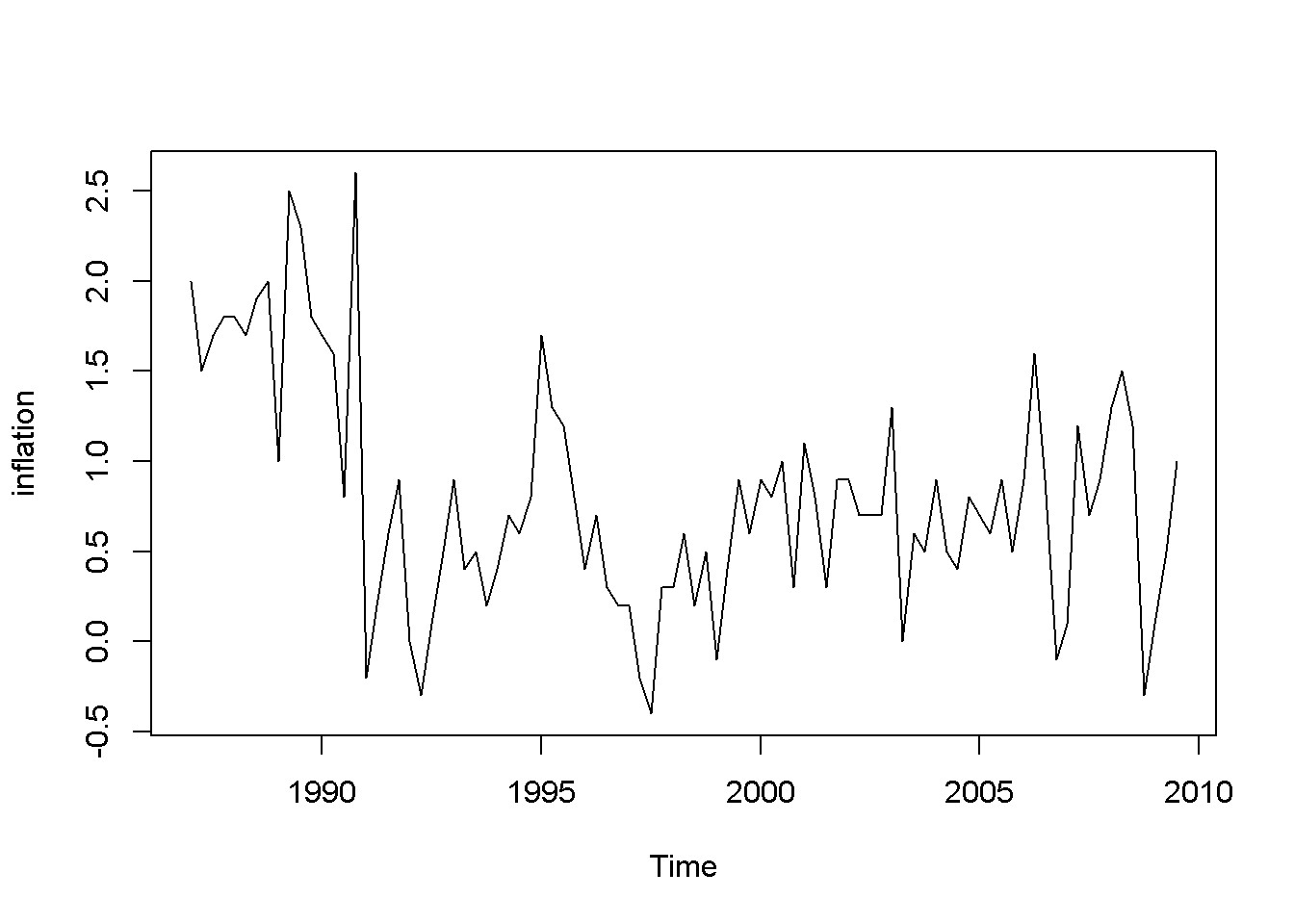


Figure 9.4: Data time plots in the ’phillips’ dataset

Table 9.5: Summary of the ‘phillips‘ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.777621 | 0.065825 | 11.81347 | 0.000000 |
| diff(u) | -0.527864 | 0.229405 | -2.30101 | 0.023754 |

9.4 satisfies the non-autocorrelation assumption of the time series regression model.

|  |
| --- |
| phill.dyn <- **dynlm**(inf~**diff**(u),data=phill.ts)  ehat <- **resid**(phill.dyn)  **kable**(**tidy**(phill.dyn), caption="Summary of the `phillips` model") |

Table 9.5 gives a *p*-value of 0*.*0238, which is significant at a 5% level. But is this *p*-value reliable? Let us investigate the autocorrelation structure of the errors in this model.

**plot**(ehat)

**abline**(h=0, lty=2)

corrgm <- **acf**(ehat)

**plot**(corrgm)

The time series plot in Figure 9.5 suggests that some patterns exists in the residuals, which is confirmed by the correlogram in Figure 9.6. The previous result suggesting that there is a significant relationship between inflation and change in unemployment rate may not be, afterall, too reliable.

While visualising the data and plotting the correlogram are powerful methods of spotting autocorrelation, in many applications we need a precise criterion, a test statistic to decide whether autocorrelation is a problem. One such a method is the **Lagrange Multiplier** test. Suppose we want to test for autocorrelation in the residuals the model given in Equation 9.5, where we assume that the errors have the autocorrelation structure described in Equation 9.6.

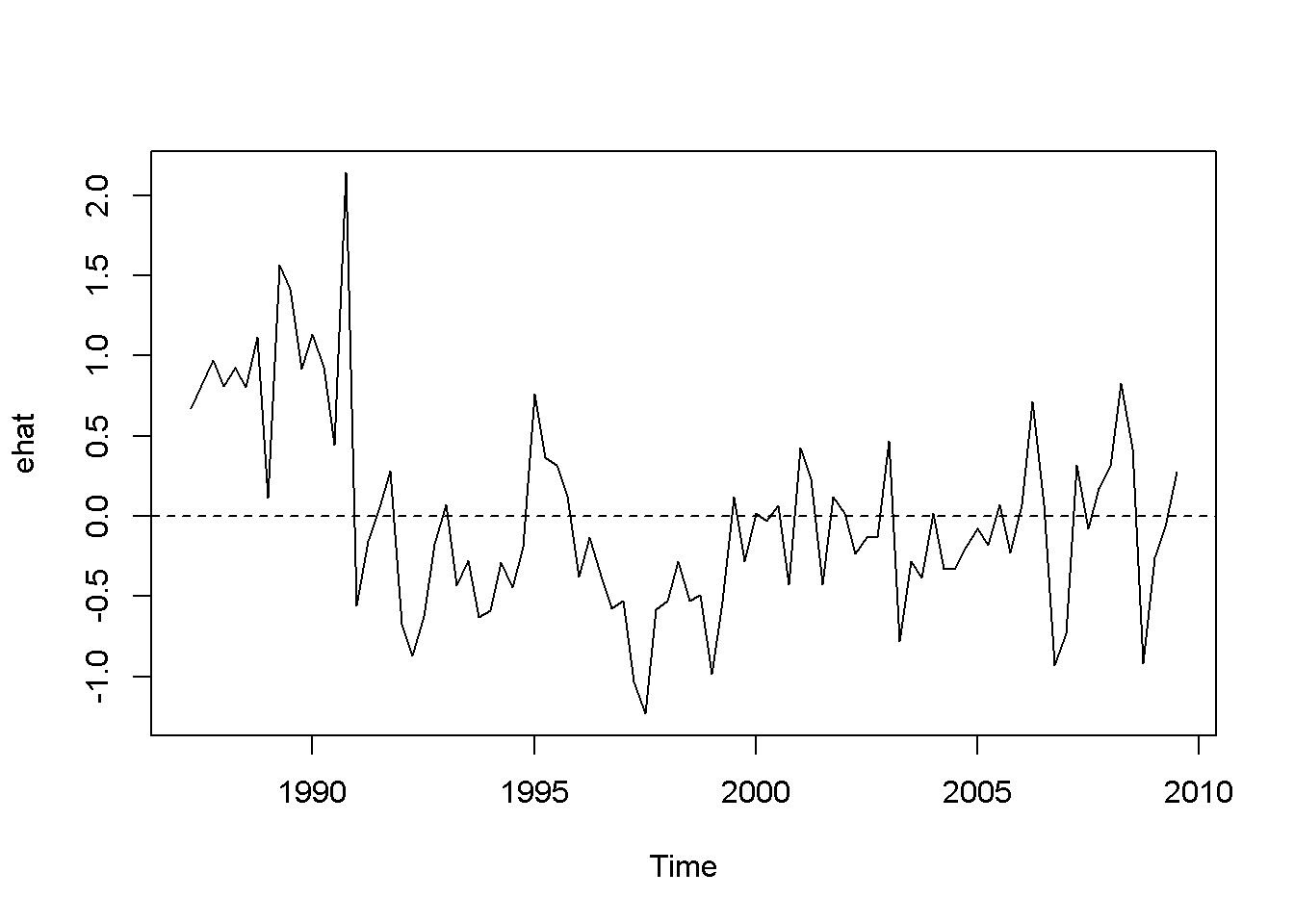


Figure 9.5: Residuals of the Phillips equation

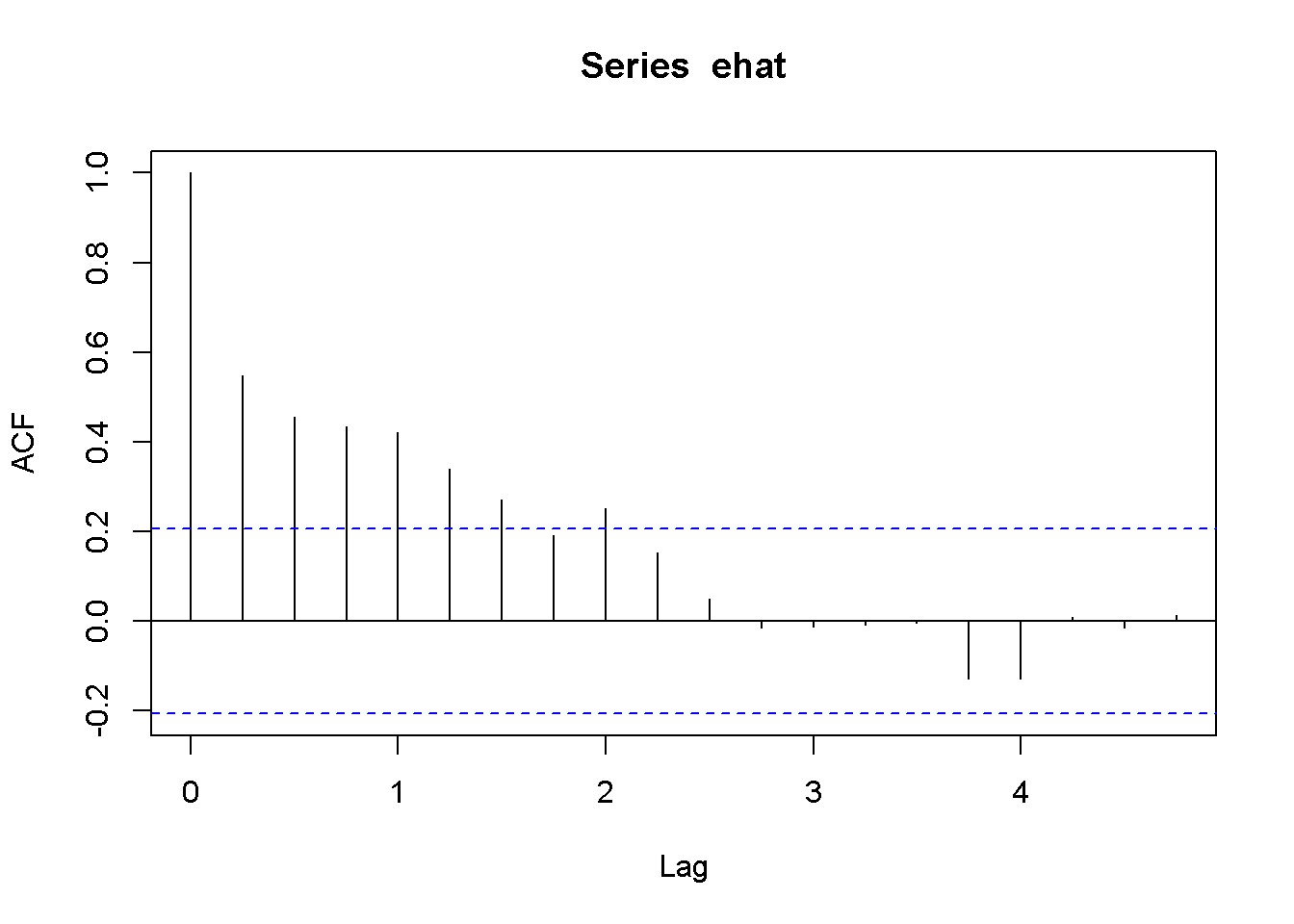


Figure 9.6: Correlogram of the residuals in the Phillips model

|  |  |
| --- | --- |
| *yt* = *β*1 + *β*2*xt* + *et* | (9.5) |
| *et* = *ρet−*1 + *νt* | (9.6) |

A test for autocorrelation would be based on the hypothesis in Equation 9.7.

A close up of a number

Description automatically generated (9.7)

After little algebraic manipulation, the auxiliary regression that the LM test actually uses is the one in Equation 9.8.

*e*ˆ*t* = *γ*1 + *γ*2*xt* + *ρe*ˆ*t−*1 + *νt* (9.8)

The test statistic is *T × R*2, where *R*2 is the coefficient of determination resulted from estimating the auxiliary equation (Equation 9.8). In *R*, all these calculations can be done in one command, bgtest(), which is the **Breusch-Godfrey** test for autocorrelation function. This function can test for autocorrelation of higher orders, which requires including higher lags for *e*ˆ in the auxiliary equation.

Let us do this test for the Phillips example. The next code sequence does the test first for only one lag and using an *F*-statistic; then, for lags up to 4, using a *χ*2-statistic. *R* does this test by either eliminating the first observations that are necessary to calculate the lags (fill=NA), or by setting them equal to zero (fill=0).

|  |
| --- |
| a <- **bgtest**(phill.dyn, order=1, type="F", fill=0)  b <- **bgtest**(phill.dyn, order=1, type="F", fill=NA)  c <- **bgtest**(phill.dyn, order=4, type="Chisq", fill=0)  d <- **bgtest**(phill.dyn, order=4, type="Chisq", fill=NA)  dfr <- **data.frame**(**rbind**(a[**c**(1,2,4)],  b[**c**(1,2,4)], c[**c**(1,2,4)], d[**c**(1,2,4)] ))  dfr <- **cbind**(**c**("1, F, 0",  "1, F, NA", "4, Chisq, 0", "4, Chisq, NA"), dfr)  **names**(dfr)<-**c**("Method", "Statistic", "Parameters", "p-Value")  **kable**(dfr, caption="Breusch-Godfrey test for the Phillips example") |

Table 9.6: Breusch-Godfrey test for the Phillips example

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Statistic | Parameters | p-Value |
| 1, F, 0 | 38.4654 | 1, 87 | 1.82193e-08 |
| 1, F, NA | 38.6946 | 1, 86 | 1.73442e-08 |
| 4, Chisq, 0 | 36.6719 | 4 | 2.10457e-07 |
| 4, Chisq, NA | 33.5937 | 4 | 9.02736e-07 |

All the four tests summarized in Table 9.6 reject the null hypothesis of no autocorrelation. The above code sequence requires a bit of explanation. The first four lines do the BG test on the previously estimated Phillips model (phill.dyn), using various parametrs: order tells *R* how many lags we want; type gives the test statistic to be used, either *F* or *χ*2; and, finally, fill tells *R* whether to delete the first observations or replace them with *e* = 0. The first column in Table 9.6 summarizes these options: number of lags, test statistic used, and how the missing observations are handeled.

The remaining of the code sequence is just to create a clear and convenient way of presenting the results of the four tests. As always, to inspect the content of an object like a type and run the command names(object).

How many lags should be considered when performing the autocorrelation test? One suggestion would be to limit the test to the number of lags that the correlogram shows to exceed the confidence band.

*R* can perform another autocorrelation test, **Durbin-Watson**, which is being used less and less today because of its limitations. However, it may be considered when the sample is small. The following command can be used to perform this test:

**dwtest**(phill.dyn)

##

## Durbin-Watson test

##

## data: phill.dyn

## DW = 0.8873, p-value = 2.2e-09

## alternative hypothesis: true autocorrelation is greater than 0

Lecture 2 -Estimation with Serially Correlated Errors

Similar to the case of heteroskedasticity, autocorrelation in the errors does not produce biased estimates of the coefficients in linear regression, but it produces incorrect standard errors. The similarity with heteroskedasticity goes even further: with autocorrelation it is possible to calculate correct (**heteroskedasticity and autocorrelation consistent, HAC**) standard errors, known as **Newey-West** standard errors.

Table 9.7: Comparing standard errors for the Phillips model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Incorrect | vcovHAC | NeweyWest | kernHAC |
| (Intercept | 0.066 | 0.095 | 0.128 | 0.131 |
| Du | 0.229 | 0.304 | 0.331 | 0.335 |

Table 9.7 compares three versions of HAC standard errors for the Phillips equation plus the incorrect ones from the initial equation. The differences come from different choices for the methods of calculating them.

There are several functions in *R* that compute HAC standard errors, of which I choose three, all available in the package sandwich.

|  |
| --- |
| **library**(sandwich)  s0 <- **coeftest**(phill.dyn)  s1 <- **coeftest**(phill.dyn, vcov.=**vcovHAC**(phill.dyn))  s2 <- **coeftest**(phill.dyn, vcov.=**NeweyWest**(phill.dyn))  s3 <- **coeftest**(phill.dyn, vcov.=**kernHAC**(phill.dyn))  tbl <- **data.frame**(**cbind**(s0[**c**(3,4)],s1[**c**(3,4)], s2[**c**(3,4)],s3[**c**(3,4)]))  **names**(tbl) <- **c**("Incorrect","vcovHAC", "NeweyWest", "kernHAC") **row.names**(tbl) <- **c**("(Intercept", "Du")  **kable**(tbl, digits=3,  caption="Comparing standard errors for the Phillips model") |

Correcting the standard errors in a model with autocorrelated errors does not make the estimator of the coefficients a minimum-variance one. Therefore, we would like to find better estimators, as we did in the case of heteroskedasticity. Let us look at models with a particular structure of the error term , a structure called **first-order autoregressive process**, or **AR(1)model**, described in Equation 9.9. The variable in this process is assumed to have zero mean and constant variance, *σν*2, and the errors *νt* should not be autocorrelated. In addition, the **autocorrelation coefficient**, *ρ*, should take values in the interval (*−*1*,*1). It can be shown that *ρ* = *corr*(*et,et−*1).

*et* = *ρet−*1 + *νt* (9.9)

The following code lines calculate and display the correlation coefficients for the first five lags in the residuals of the Phillips equation (Equation 9.4). Please notice that the autocorrelations tend to become smaller and smaller for distant lags, but they still remain higher than Equation 9.9 implies.

|  |
| --- |
| ehat <- **resid**(phill.dyn)  ac <- **acf**(ehat, plot=FALSE)  *# The Phillips equation: five lag correlations in residuals*  ac$acf[2:6] |

## [1] 0.548659 0.455732 0.433216 0.420494 0.339034

The correlation coefficient for the first lag is an estimate of the coefficient *ρ* in the *AR*(1) process defined in Equation 9.9, *ρ*ˆ1 = *ρ*ˆ = *r*1 = 0*.*549

## 9.5 Nonlinear Least Squares Estimation

The simple linear regression model in Equation 9.5 with AR(1) errors defined in Equation 9.6 can be transformed into a model having uncorrelated errors, as Equation 9.10 shows.

*yt* = *β*1(1 *− ρ*) + *β*2*xt* + *ρyt−*1 *− ρβ*2*xt−*1 + *νt* (9.10)

Equation 9.10 is nonlinear in the coefficients, and therefore it needs special methods of estimation. Applying the same transformations to the Phillips model given in Equation 9.4, we obtain its nonlinear version, Equation 9.11.

*inft* = *β*1(1 *− ρ*) + *β*2*Dut* + *ρinft−*1 *− ρβ*2*Dut−*1 + *νt* (9.11)

The next code line estimates the non-linear model in Equation 9.11 using the nls() function, which requires the data under a data frame form. The first few lines of code create a separate variables for inf and u and their lags, then brings all of them together in a data frame. The main arguments of the nls function are the following: formula, a nonlinear function of the regression parameters, data= a data frame, and ‘start=list(initial guess values of the parameters), and others.

|  |
| --- |
| **library**(dynlm)  phill.dyn <- **dynlm**(inf~**diff**(u), data=phill.ts)  *# Non-linear AR(1) model with 'Cochrane-Orcutt method'nls'*  phill.ts.tab <- **cbind**(phill.ts[,"inf"],  phill.ts[,"u"],  **lag**(phill.ts[,"inf"], -1),  **diff**(phill.ts[,"u"], lag=1), |
| **lag**(**diff**(phill.ts[,2],lag=1), -1) )  phill.dfr <- **data.frame**(phill.ts.tab)  **names**(phill.dfr) <- **c**("inf", "u", "Linf", "Du", "LDu")  phill.nls <- **nls**(inf~b1\*(1-rho)+b2\*Du+rho\*Linf-  rho\*b2\*LDu,  data=phill.dfr,  start=**list**(rho=0.5, b1=0.5, b2=-0.5))  s1 *# This is `phill.dyn` with HAC errors:* |

##

## t test of coefficients:

##

##Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.77762 0.09485 8.198 1.82e-12 \*\*\* ## diff(u) -0.52786 0.30444 -1.734 0.0864 .

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

phill.dyn *# The simple linear model:*

##

## Time series regression with "ts" data:

## Start = 1987(2), End = 2009(3)

##

## Call:

## dynlm(formula = inf ~ diff(u), data = phill.ts)

##

## Coefficients:

## (Intercept) diff(u) ## 0.778 -0.528

phill.nls *# The 'nls' model:*

## Nonlinear regression model

## model: inf ~ b1 \* (1 - rho) + b2 \* Du + rho \* Linf - rho \* b2 \* LDu

## data: phill.dfr

## rho b1 b2 ## 0.557 0.761 -0.694

## residual sum-of-squares: 23.2

##

## Number of iterations to convergence: 3

## Achieved convergence tolerance: 8.06e-06

**coef**(phill.nls)[["rho"]]

## [1] 0.557398

Comparing the nonlinear model with the two linear models (with, and without HAC standard errors) shows differences in both coefficients and standard errors. This is an indication that nonlinear estimation is a better choice than HAC standard errors. Please note that the NLS model provides an estimate of the autocorrelation coefficient, *ρ* = 0*.*557.

## 9.6 A More General Model

Equation 9.12 gives an **autoregresive distributed lag** model, which is a generalization of the model presented in Equation 9.10. The two models are equivalent under the restriction *δ*1 = *−θ*1*δ*0.

*yt* = *δ* + *θ*1*yt−*1 + *δ*0*xt* + *δ*1*xt−*1 + *νt* (9.12)

Equation 9.13 is the Phillips version of the ARDL model given by Equation 9.12.

*inft* = *δ* + *θ*1*inft−*1 + *δ*0*Dut* + *δ*1*Dut−*1 + *νt* (9.13)

A Wald test can be used to decide if the two models, the nonlinear one and the more general one are equivalent.

s.nls <- **summary**(phill.nls)

phill.gen <- **dynlm**(inf~**L**(inf)+**d**(u)+**L**(**d**(u)), data=phill.ts)

s.gen <- **summary**(phill.gen)

nlW <- **nlWaldtest**(phill.gen, texts="b[4]=-b[2]\*b[3]")

The *R* function performing the Wald test is nlWaldtest in package nlWaldTest, which can test nonlinear restrictions. The result in our case is a *χ*2 value of 0*.*11227, with *p*-value of 0*.*737574, which does not reject the null hypothesis that the restriction *δ*1 = *−θ*1*δ*0 cannot be rejected, making, in turn, Equations 9.11 and 9.13 equivalent.

The code lines above use the L and d from package dynlm for constructing lags and differences in time series. Unlike the similar functions that we have previously used (lag and diff), these do not work outside the command dynlm. Please note that constructing lags with lag() requires specifying the negative sign of the lag, which is not necessary for the L() function.

Table 9.8: Using dynlm with L and d operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.333633 | 0.089903 | 3.71104 | 0.000368 |
| L(inf) | 0.559268 | 0.090796 | 6.15959 | 0.000000 |
| d(u) | -0.688185 | 0.249870 | -2.75417 | 0.007195 |
| L(d(u)) | 0.319953 | 0.257504 | 1.24252 | 0.217464 |

Table 9.9: Using dynlm with lag and diff operators

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | Estimate | std.error | statistic | p.value |
| (Intercept) | 0.333633 | 0.089903 | 3.71104 | 0.000368 |
| lag(inf, -1) | 0.559268 | 0.090796 | 6.15959 | 0.000000 |
| diff(u) | -0.688185 | 0.249870 | -2.75417 | 0.007195 |
| lag(diff(u), -1) | 0.319953 | 0.257504 | 1.24252 | 0.217464 |

phill1.gen <- **dynlm**(inf~**lag**(inf, -1)+**diff**(u)+**lag**(**diff**(u), -1), data=phill.ts)

The results can be compared in Tables 9.8 and 9.9.

**kable**(**tidy**(phill.gen), caption="Using dynlm with L and d operators")

**kable**(**tidy**(phill1.gen), caption="Using dynlm with lag and diff operators")

## 9.7 Autoregressive Models

An autoregressive model of order *p*, *AR*(*p*) (Equation 9.14) is a model with *p* lags of the response acting as independent variables and no other regressors.

*yt* = *δ* + *θ*1*yt−*1 + *θ*2*yt−*2 + *...* + *θpyt−p* + *νt* (9.14)

The next code sequence and Table 9.10 show an autoregressive model of order 2 using the series *g* in the data file *okun*.

|  |
| --- |
| **data**(okun)  okun.ts <- **ts**(okun)  okun.ar2 <- **dynlm**(g~**L**(g)+**L**(g,2), data=okun.ts) |

Table 9.10: Autoregressive model of order 2 using the dataset *okun*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | Estimate | std.error | statistic | p.value |
| (Intercept) | 0.4657 | 0.1433 | 3.2510 | 0.0016 |
| L(g) | 0.3770 | 0.1000 | 3.7692 | 0.0003 |
| L(g, 2) | 0.2462 | 0.1029 | 2.3937 | 0.0187 |



Figure 9.7: Residual correlogram for US GDP AR(2) model

**kable**(**tidy**(okun.ar2), digits=4, caption="Autoregressive model of order 2 using the dataset $okun$")

How can we decide how many lags to include in an autoregressive model? One way is to look at the correlogram of the series and include the lags that show autocorrelation. The follwing code creates the correlogram in the AR(2) residuals of the US GDP series. The correlogram in Figure 9.7 shows that basically only the first two lags may be included.

res.ar2 <- **resid**(okun.ar2) **Acf**(res.ar2, lag.max=12) *# New: Acf() from package forecast*

Another way of choosing the specification of an autoregressive model is to compare models of several orders and choose the one that provides the lowest AIC or BIC value.

Table 9.11: Lag order selection for an AR model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| AIC | 169.4 | 163.5 | 163.1 | 161.6 | 162.5 |
| BIC | 177.1 | 173.8 | 175.9 | 176.9 | 180.3 |

|  |
| --- |
| aics <- **rep**(0,5)  bics <- **rep**(0,5) y <- okun.ts[,"g"]  for (i in 1:5){  ari <- **dynlm**(y~**L**(y,1:i), start=i)  aics[i] <- **AIC**(ari)  bics[i] <- **BIC**(ari)  }  tbl <- **data.frame**(**rbind**(aics, bics)) **names**  (tbl) <- **c**("1","2","3","4","5") **row.names**  (tbl) <- **c**("AIC","BIC")  **kable**(tbl, digits=1, align='c', caption="Lag order selection for an AR model") |

Table 9.11 displays the AIC and BIC (or SC) values for autoregressive models on the US GDP in dataset *okun* with number of lags from 1 to 5. (As mentioned before, the numbers do not coincide with those in PoE, but the ranking does.) The lowest AIC value indicates that the optimal model should include four lags, while the BIC values indicate the model with only two lags as the winner. Other criteria may be taken into account in such a situation, for instance the correlogram, which agrees with the BIC’s choice of model.

The previous code fragment needs some explanation. The first two lines initialize the vectors that are going to hold the AIC and BIC results. The problem of automatically changing the number of regressors is addressed by using the convenient function L(y, 1:i). AIC() and BIC() are basic *R* functions.

## 9.8 Forecasting

Let us consider first an autoregressive (AR) model, exemplified by the US GDP growth with two lags specified in Equation 9.15.

*gt* = *δ* + *θ*1*gt−*1 + *θ*2*gt−*2 + *νt* (9.15)

Table 9.12: The AR(2) growth model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.465726 | 0.143258 | 3.25097 | 0.001602 |
| L(y, 1:2)1 | 0.377001 | 0.100021 | 3.76923 | 0.000287 |
| L(y, 1:2)2 | 0.246239 | 0.102869 | 2.39372 | 0.018686 |

Table 9.13: Forecasts for the AR(2) growth model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Point.Forecast | Lo.80 | Hi.80 | Lo.95 | Hi.95 |
| 99 | 0.718 | 0.021 | 1.415 | -0.348 | 1.784 |
| 100 | 0.933 | 0.188 | 1.678 | -0.206 | 2.073 |
| 101 | 0.994 | 0.202 | 1.787 | -0.218 | 2.207 |

Once the coefficients of this model are estimated, they can be used to predict (forecast) out-of-sample, future values of the response variable. Let us do this for periods *T* + 1, *T* + 2, and *T* + 3, where *T* is the last period in the sample. Equation 9.16 gives the forecast for period *s* into the future.

*gT*+*s* = *δ* + *θ*1*gT*+*s−*1 + *θ*2*gT*+*s−*2 + *νT*+*s* (9.16)

y <- okun.ts[,"g"]

g.ar2 <- **dynlm**(y~**L**(y, 1:2))

**kable**(**tidy**(g.ar2), caption="The AR(2) growth model")

Table 9.12 shows the results of the AR(2) model. *R* uses the function forecast() in package forecast to automatically calculate forecasts based on autoregressive or other time series models. One such model is ar(), which fits an autoregressive model to a univariate time series. The arguments of ar() include: x= the name of the time series, aic=TRUE, if we want automatic selection of the number of lags based on the AIC information criterion; otherwise, aic=FALSE; order.max= the maximum lag order in the autoregressive model.

ar2g <- **ar**(y, aic=FALSE, order.max=2, method="ols")

fcst <- **data.frame**(**forecast**(ar2g, 3))

**kable**(fcst, digits=3, caption="Forecasts for the AR(2) growth model")

Table 9.13 shows the forecasted values for three future periods based on the AR(2) growth model.

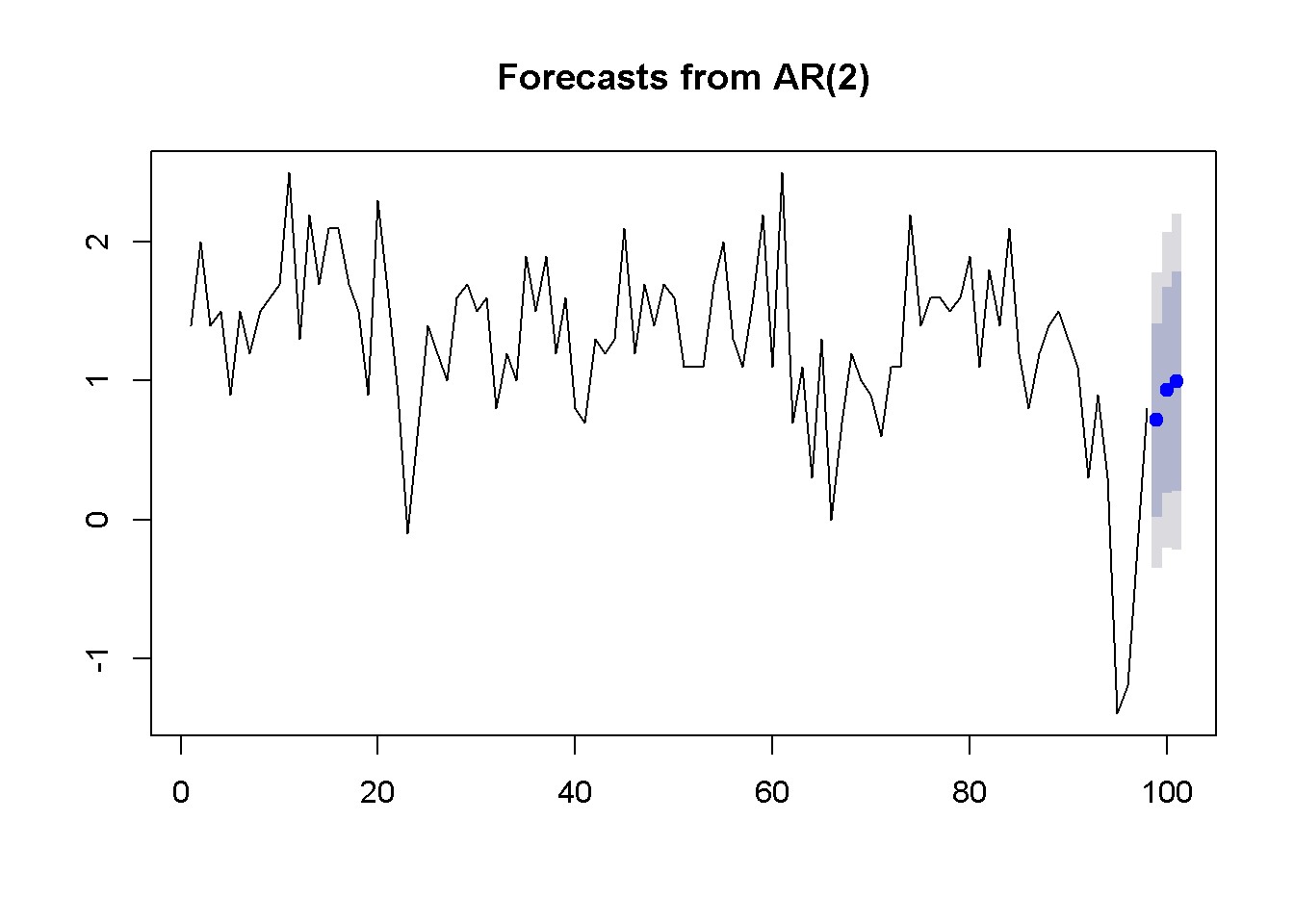


Figure 9.8: Forecasts and confidence intervals for three future periods

**plot**(**forecast**(ar2g,3))

Figure 9.8 illustrates the forecasts produced by the AR(2) model of US GDP and their interval estimates.

Using more information under the form of additional regressors can improve the accuracy of forecasting. We have already studied ARDL models; let us use such a model to forecast the rate of unemployment based on past unemployment and GDP growth data. The dataset is still *okun*, and the model is an ARDL(1,1) as the one in Equation 9.17.

A number and plus symbol

Description automatically generated with medium confidence(9.17)

Since we are interested in forecasting *levels* of unemployment, not differences, we want to transform the ARDL(1,1) model in Equation 9.17 into the ARDL(2,1) one in Equation 9.18.

A math equation with black text

Description automatically generated with medium confidence(9.18)

Another forecasting model is the **exponential smoothing** one, which, like the AR model, uses only the variable to be forecasted. In addition, the exponential smoothing model requires a weighting parameter, *α*, and an initial level of the forecasted variable, *y*ˆ. Next period’s forecast is a weighted average of this period’s level and this period’s forecast, as indicated in Equation 9.19.

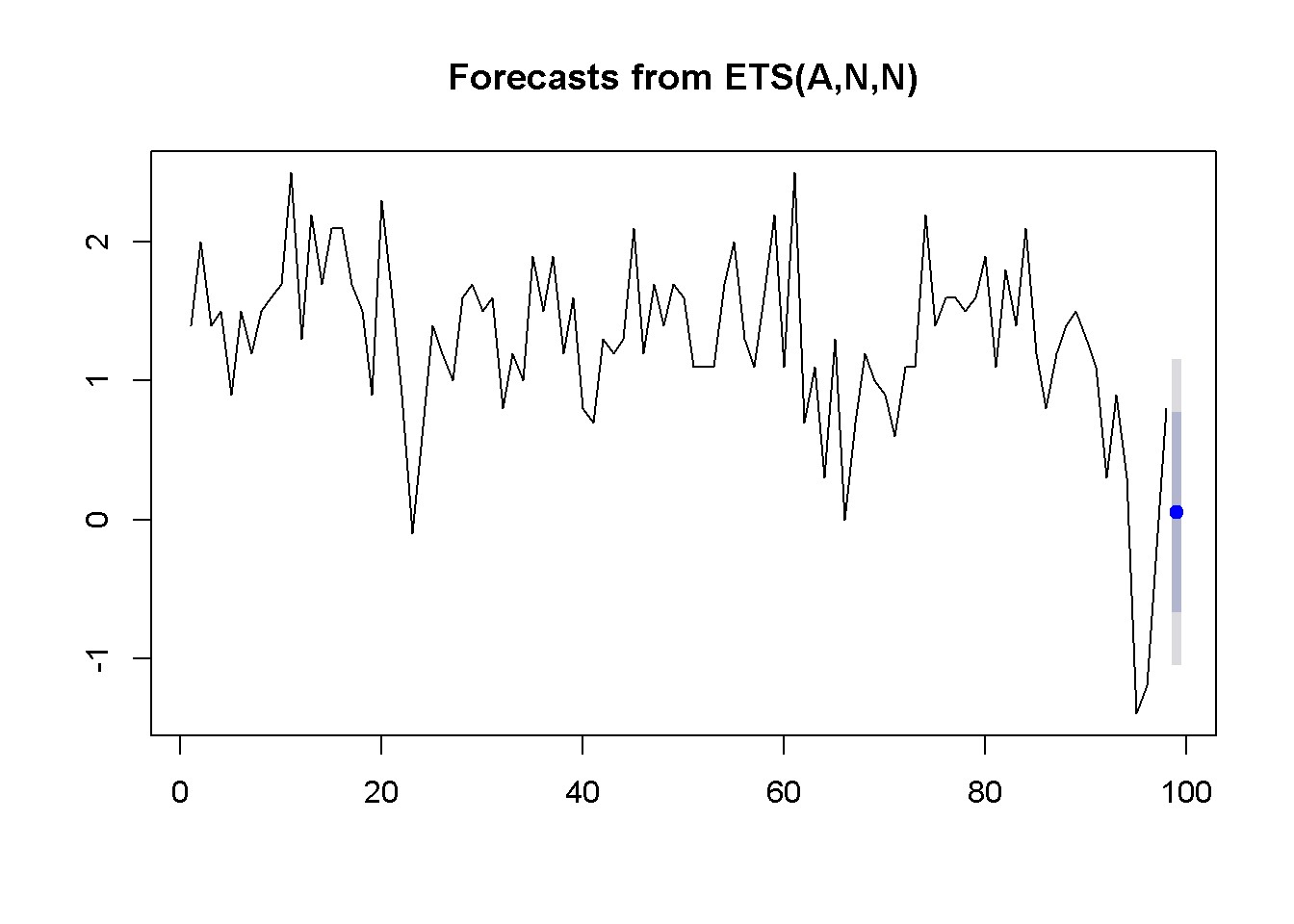


Figure 9.9: Exponential smoothing forecast using ’ets’

*y*ˆ*T*+1 = *αyT* + (1 *− α*)*y*ˆ*T* (9.19)

There are sevaral ways to compute an exponential smoothing forecast in *R*. I only use two here, for comparison. The first is the function ets() in the forecast package.

|  |
| --- |
| y <- okun.ts[,"g"]  okun.ets <- **ets**(y)  okunf.ets <- **forecast**(okun.ets,1) *#one-period forecast*  **plot**(okunf.ets) |

The results include the computed value of the weight, *α* = 0*.*381, the point estimate of the growth rate *gT*+1 = 0*.*054, and the 95 percent interval estimate (*−*1*.*051*,*1*.*158). Figure 9.9 illustrates these results.

The second method uses the function HoltWinters and is shown in the following code fragment.

okun.HW <- **HoltWinters**(y, beta=FALSE, gamma=FALSE)

**plot**(okun.HW)

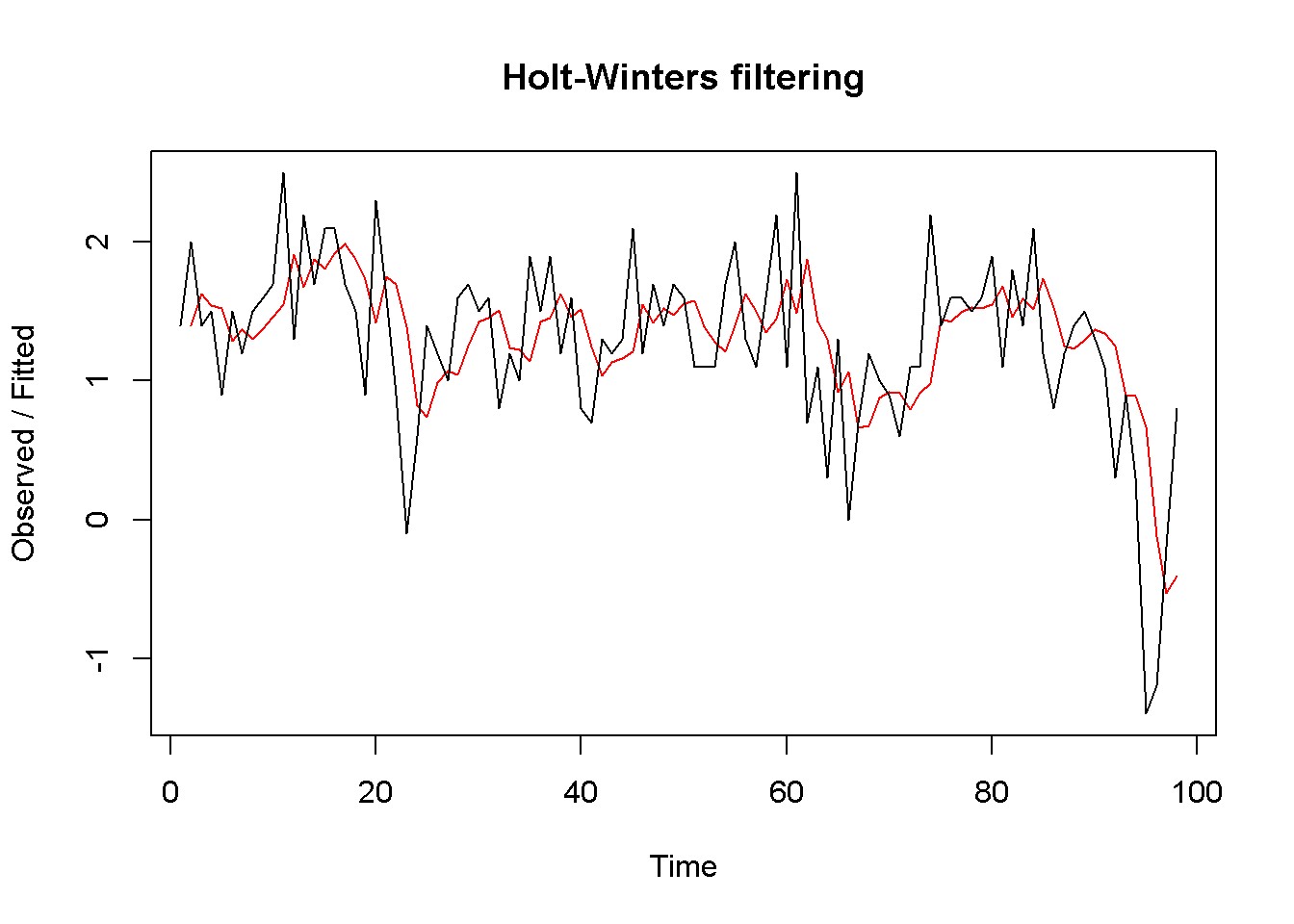


Figure 9.10: Exponential smoothing forecast using ’HoltWinters’

okunf.HW <- **forecast**(okun.HW,1)

Similar to the ets function, HoltWinters automatically determines the optimal weight *α* of the exponential smoothing model. In our case, the value is *α* = 0*.*381. The point estimate is *gT*+1 = 0*.*054, and the 95 percent interval estimate is (*−*1*.*06*,*1*.*168). Figure 9.10 compares the forecasted with the actual series.

## 9.9 Multiplier Analysis

In an *ARDL*(*p,q*) model, if one variable changes at some period it affects the response over several subsequent periods. Multiplier analysis quantifies these time effects. Let me remind you the form of the *ARDL*(*p,q*) model in Equation 9.20.

*yt* = *δ* + *θ*1*yt−*1 + *...* + *θpyt−p* + *δ*0*xt* + *δ*1*xt−*1 + *...* + *δqxt−q* + *νt* (9.20)

The model in Equation 9.20 can be transformed by iterative substitution in an infinite distributed lag model, which includes only explanatory variables with no lags of the response. The transformed model is shown in Equation 9.21.

*yt* = *α* + *β*0*xt* + *β*1*xt−*1 + *β*2*xt−*2 + *β*3*xt−*3 + *...* + *et* (9.21)

Coefficient *βs* in Equation 9.21 is called the *s-period delay multiplier*; the sum of the *β*s from today to period *s* in the past is called *interim multiplier*, and the sum of all periods to infinity is called *total multiplier*.

Video 2 - Random Regressors

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!*  **library**(AER) *#for `ivreg()`*  **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`*  **library**(PoEdata) *#for PoE4 datasets*  **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for making neat tables with `kable()`*  **library**(stargazer) |

In most data coming from natural, non-controlled phenomena, both the dependent and independent variables are random. In many cases, some of the independent variables are also correlated with the error term in a regression model, which makes the OLS method inappropriate. Regressors (*x*) that are correlated with the error term are called **endogeneous**; likewise, those that are not are called **exogeneous**. The remedy for this violation of the linear regression assumption is the use of **instrumental variables**, or **instruments**, which are variables (*z*) that do not directly influence the response but are correlated with the endogenous regressor in question.

## 10.1 The Instrumental Variables (IV) Method

A *strong* instrument, one that is highly correlated with the endogenous regressor it concerns, reduces the variance of the estimated coefficient. Assume the multiple regression model in Equation 10.1, where regressors *x*2 to *xK−*1 are exogenous and *xK* is endogenous. The IV method consists in two stages: first regress *xK* on all the other regressors and all the instruments and create the fitted values series, *x*ˆ*K*; second, regress the initial equation, in which *xK* is replaced by *x*ˆ*K*. Therefore, the IV method is often called **two-stage least squares**, or **2SLS**.

Table 10.1: First stage in the 2SLS model for the ’wage’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 9.7751 | 0.4239 | 23.0605 | 0.0000 |
| exper | 0.0489 | 0.0417 | 1.1726 | 0.2416 |
| I(exper^2) | -0.0013 | 0.0012 | -1.0290 | 0.3040 |
| mothereduc | 0.2677 | 0.0311 | 8.5992 | 0.0000 |

*y* = *β*1 + *β*2*x*2 + *...* + *βKxK* + *e* (10.1)

Consider the *wage* model in Equation 10.2 using the *mroz* dataset. The notorious difficulty with this model is that the error term may include some unobserved attributes, such as personal ability, that determine both wage and education. In other words, the independent variable *educ* is correlated with the error term, is

*endogenous*.

*log*(*wage*) = *β*1 + *β*2*educ* + *β*3*exper* + *β*4*exper*2 + *e* (10.2)

An instrument that may address the endogeneity of *educ* is *mothereduc*, of which we can reasonably assume that it does not directly influence the daughter’s wage, but it influences her education.

Let us first carry out an explicit two-stage model with only one instrument, *mothereduc*. The first stage is to regress *educ* on other regressors and the instrument, as Equation 10.3 shows.

*educ* = *γ*1 + *γ*2*exper* + *γ*3*exper*2 + *θ*1*mothereduc* + *νeduc* (10.3)

|  |
| --- |
| **data**("mroz", package="PoEdata")  mroz1 <- mroz[mroz$lfp==1,] *#restricts sample to lfp=1* educ.ols <- **lm**(educ~exper+**I**(exper^2)+mothereduc, data=mroz1) **kable**(**tidy**(educ.ols), digits=4, align='c',caption=  "First stage in the 2SLS model for the 'wage' equation") |

The *p*-value for *mothereduc* is very low (see Table 10.1), indicating a strong correlation between this instrument and the endogenous variable *educ* aven after controling for other variables. The second stage in the two-stage procedure is to create the fitted values of *educ* from the first stage (Equation 10.3) and plug them into the model of interest, Euation 10.2 to replace the original variable *educ*.Table 10.2: Second stage in the 2SLS model for the ’wage’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.1982 | 0.4933 | 0.4017 | 0.6881 |
| educHat | 0.0493 | 0.0391 | 1.2613 | 0.2079 |
| exper | 0.0449 | 0.0142 | 3.1668 | 0.0017 |
| I(exper^2) | -0.0009 | 0.0004 | -2.1749 | 0.0302 |
| educHat <- **fitted**(educ.ols)  wage.2sls <- **lm**(**log**(wage)~educHat+exper+**I**(exper^2), data=mroz1) **kable**(**tidy**(wage.2sls), digits=4, align='c',caption=  "Second stage in the 2SLS model for the 'wage' equation") | | | | | | |

The results of the explicit 2*SLS* procedure are shown in Table 10.2; keep n mind, however, that the standard errors calculated in this way are incorrect; the correct method is to use a dedicated software function to solve an instrumental variable model. In *R*, such a function is ivreg().

|  |
| --- |
| **data**("mroz", package="PoEdata")  mroz1 <- mroz[mroz$lfp==1,] *#restricts sample to lfp=1.*  mroz1.ols <- **lm**(**log**(wage)~educ+exper+**I**(exper^2), data=mroz1)  mroz1.iv <- **ivreg**(**log**(wage)~educ+exper+**I**(exper^2)| exper+**I**(exper^2)+mothereduc, data=mroz1)  mroz1.iv1 <- **ivreg**(**log**(wage)~educ+exper+**I**(exper^2)| exper+**I**(exper^2)+mothereduc+fathereduc, data=mroz1)  **stargazer**(mroz1.ols, wage.2sls, mroz1.iv, mroz1.iv1,  title="Wage equation: OLS, 2SLS, and IV models compared", header=FALSE,  type=.stargazertype, *# "html" or "latex" (in index.Rmd)*  keep.stat="n", *# what statistics to print*  omit.table.layout="n", star.cutoffs=NA, digits=4,  *# single.row=TRUE,*  intercept.bottom=FALSE, *#moves the intercept coef to top*  column.labels=**c**("OLS","explicit 2SLS", "IV mothereduc",  "IV mothereduc and fathereduc"),  dep.var.labels.include = FALSE, |

model.numbers = FALSE,

dep.var.caption="Dependent variable: wage",

model.names=FALSE, star.char=NULL) *#supresses the stars)*

Table 10.3: Wage equation: OLS, 2SLS, and IV models compared

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | Dependent variable: wage | | |
| OLS | explicit 2SLS | IV mothereduc | IV mothereduc and fathereduc |
| Constant | *−*0.5220 | 0.1982 | 0.1982 | 0.0481 |
|  | (0.1986) | (0.4933) | (0.4729) | (0.4003) |
| Educ | 0.1075 |  | 0.0493 | 0.0614 |
|  | (0.0141) |  | (0.0374) | (0.0314) |
| educHat |  | 0.0493  (0.0391) |  |  |
| Exper | 0.0416 | 0.0449 | 0.0449 | 0.0442 |
|  | (0.0132) | (0.0142) | (0.0136) | (0.0134) |
| I(experˆ2) | *−*0.0008 | *−*0.0009 | *−*0.0009 | *−*0.0009 |
|  | (0.0004) | (0.0004) | (0.0004) | (0.0004) |
| Observations | 428 | 428 | 428 | 428 |

The table titled “Wage equation: OLS, 2SLS, and IV compared” shows that the importance of education in determining wage decreases in the IV model. It also shows that the explicit 2SLS model and the IV model with only *mothered* instrument yield the same coefficients (the *educ* in the IV model is equivalent to the *educHat* in 2SLS), but the standard errors are different. The correct ones are those provided by the IV model.

A few observations are in order concerning the above code sequence. First, since some of the individuals are not in the labor force, their wages are zero and the log cannot be calculated. I excluded those observations using only those for which *lpf* is equal to 1. Second, the instrument list in the command ivreg includes both the instrument itself (*mothereduc*) and all exogenous regressors, which are, so to speak, their own instruments. The vertical bar character | separates the proper regressor list from the instrument list.

Table 10.4: The ’educ’ first-stage equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 9.1026 | 0.4266 | 21.3396 | 0.0000 |
| exper | 0.0452 | 0.0403 | 1.1236 | 0.2618 |
| I(exper^2) | -0.0010 | 0.0012 | -0.8386 | 0.4022 |
| mothereduc | 0.1576 | 0.0359 | 4.3906 | 0.0000 |
| fathereduc | 0.1895 | 0.0338 | 5.6152 | 0.0000 |

To test for weak instruments in the *wage* equation, we just test the joint significance of the instruments in an *educ* model as shown in Equation 10.4. *educ* = *γ*1 + *γ*2*exper* + *γ*3*exper*2 + *θ*1*mothereduc* + *θ*2*fathereduc* + *ν* (10.4)

|  |
| --- |
| educ.ols <- **lm**(educ~exper+**I**(exper^2)+mothereduc+fathereduc, data=mroz1)  tab <- **tidy**(educ.ols)  **kable**(tab, digits=4, caption="The 'educ' first-stage equation") |

**linearHypothesis**(educ.ols, **c**("mothereduc=0", "fathereduc=0"))

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| 425 | 2219.22 | NA | NA | NA | NA |
| 423 | 1758.58 | 2 | 460.641 | 55.4003 | 0 |

The test rejects the null hypothesis that both *mothereduc* and *fathereduc* coefficients are zero, indicating that at least one instrument is strong. A rule of thumb requires to soundly reject the null hypothesis at a value of the *F*-statistic greater than 10 or, for only one instrument, a *t*-statistic greater than 3.16, to make sure that an instrument is strong.

For a model to be identified the number of instruments should be at least equal to the number of endogenous variables. If there are more instruments than endogenous variables, the model is said to be **overidentified**.

## 10.2 Specification Tests

We have seen before how to test for **weak instruments** with only one instrument. This test can be extended to several instruments. The null hypothesis is *H*0: *“All instruments are weak”*.

Since using IV when it is not necessary worsens our estimates, we would like to test whether the variables that worry us are indeed endogenous. This problem is addressed by the **Hausman test for endogeneity**, where the null hypothesis is *H*0 : *Cov*(*x,e*) = 0. Thus, rejecting the null hypothesis indicates the existence of endogeneity and the need for instrumental variables.

The test for the validity of instruments (whether the instruments are corrrelated with the error term) can only be performed for the *extra* instruments, those that are in excess of the number of endogenous variables. This test is sometimes called a test for **overidentifying restrictions**, or the **Sargan** test. The null hypothesis is that the covariance between the instrument and the error term is zero, *H*0 : *Cov*(*z,e*) = 0. Thus, rejecting the null indicates that at least one of the extra instruments is not valid.

*R* automatically performs these three tests and reports the results in the output to the ivreg function.

**summary**(mroz1.iv1, diagnostics=TRUE)

##

## Call:

## ivreg(formula = log(wage) ~ educ + exper + I(exper^2) | exper +

## I(exper^2) + mothereduc + fathereduc, data = mroz1)

##

## Residuals:

## Min 1Q Median 3Q Max

## -3.0986 -0.3196 0.0551 0.3689 2.3493

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|) ## (Intercept) 0.048100 0.400328 0.12 0.9044

## educ 0.061397 0.031437 1.95 0.0515 .

## exper 0.044170 0.013432 3.29 0.0011 \*\*

## I(exper^2) -0.000899 0.000402 -2.24 0.0257 \*

##

## Diagnostic tests:

## df1 df2 statistic p-value

## Weak instruments 2 423 55.40 <2e-16 \*\*\* ## Wu-Hausman 1 423 2.79 0.095 .

## Sargan 1 NA 0.38 0.539

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.675 on 424 degrees of freedom

## Multiple R-Squared: 0.136, Adjusted R-squared: 0.13

## Wald test: 8.14 on 3 and 424 DF, p-value: 0.0000279

The results for the wage equation are as follows:

* Weak instruments test: rejects the null, meaning that at least one instrument is strong
* (Wu-)Hausman test for endogeneity: barely rejects the null that the variable of concern is uncorrelated with the error term, indicating that *educ* is marginally endogenous
* Sargan overidentifying restrictions: does not reject the null, meaning that the extra instruments are valid (are uncorrelated with the error term).

The test for weak instruments might be unreliable with more than one endogenous regressor, though, because there is indeed one *F*-statistic for each endogenous regressor. An alternative is the **Cragg-Donald** test based on the statistic shown in Equation 10.5, where *G* is the number of exogenous regressors, *B* is the number of endogenous regressors, *L* is the number of external instruments, and *rB* is the lowest **canonical correlation** (a measure of the correlation between the endogenous and the exogenous variables, calculated by the function cancor() in *R*).

A black text on a white background

Description automatically generated (10.5)

Let us look at the *hours* equation with two endogenous variables, *mtr* and *educ*, and two external instruments, *mothereduc* and *fathereduc*. One of the two exogenous regressors, *nwifeinc*, is the family income net of the wife’s income; the other exogenous regressor, *mtr*, is the wife’s marginal tax rate. Equation 10.6 shows this model; the dataset is *mroz*, restricted to women that are in the labor force.

*hours* = *β*1 + *β*2*mtr* + *β*3*educ* + *β*4*kidsl*6 + *β*5*nwifeinc* + *e* (10.6)

The next code sequence uses the *R* function cancor() to calculate the lowest of two canonical correlations, *rB*, which is needed for the Cragg-Donald *F*-statistic in Equation 10.5.

|  |
| --- |
| **data**("mroz", package="PoEdata")  mroz1 <- mroz[**which**(mroz$wage>0),]  nwifeinc <- (mroz1$faminc-mroz1$wage\*mroz1$hours)/1000  G<-2; L<-2; N<-**nrow**(mroz1)  x1 <- **resid**(**lm**(mtr~kidsl6+nwifeinc, data=mroz1))  x2 <- **resid**(**lm**(educ~kidsl6+nwifeinc, data=mroz1))  z1 <-**resid**(**lm**(mothereduc~kidsl6+nwifeinc, data=mroz1))  z2 <-**resid**(**lm**(fathereduc~kidsl6+nwifeinc, data=mroz1)) |
| X <- **cbind**(x1,x2)  Y <- **cbind**(z1,z2)  rB <- **min**(**cancor**(X,Y)$cor)  CraggDonaldF <- ((N-G-L)/L)/((1-rB^2)/rB^2) |

The result is the Cragg-Donald *F* = 0*.*100806, which is much smaller than the critical value of 4*.*58 given in Table 10E.1 of the textbook (Hill, Griffiths, and Lim 2011). This test rejects the null hypothesis of strong instruments, contradicting my previous result.

Lab 1 - Simultaneous Equations Models

**rm**(list=**ls**()) *#Removes all items in Environment!*

**library**(systemfit)

**library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 dataset* **library**(knitr) *#for kable()*

New package: systemfit (Henningsen and Hamann 2015).

Simultaneous equations are models with more than one response variable, where the solution is determined by an equilibrium among opposing forces. The econometric problem is similar to the endogenous variables we have studied already in the previous chapter because the mutual interaction between dependent variables can be considered a form of endogeneity. The typical example of an economic simultaneous equation problem is the supply and demand model, where price and quantity are interdependent and are determined by the interaction between supply and demand.

Usually, an economic model such as demand and supply equations include several of the depednedent (endogenous) variables in each equation. Such a model is called the **structural form** of the model. If the structural form is transformed such that each equation shows one dependent variable as a function of only exogenous independent variables, the new form is called the **reduced form**. The reduced form can be estimated by least squares, while the structural form cannot because it includes endogenous variables on its right-hand side.

The **necessary condition for identification** requires that, for the problem to have a solution each equation in the structural form of the system should miss at least an exogenous variable that is present in other equations.

Simultaneous equations are the object of package systemfit in *R*, with the func-

169

tion systemfit(), which requires the following main arguments: formula= a list describing the equations of the system; method= the desired (appropriate) method of estimation, which can be one of “OLS”, “WLS”, “SUR”, “2SLS”, “W2SLS”, or “3SLS” (we have only studied OLS, WLS, and 2SLS so far); inst= a list of instrumental variables under the form of one-sided model formulas; all the endogenous variables in the system must be in this list.

The following example uses the dataset *truffles*, where *q* is quantity of truffles traded, *p* is the market price, *ps* is the price of a substitute, *di* is income, and *pf* is a measure of costs of production. The structural demand and supply equations (Equations 11.1 and 11.2) are formulated based on economic theory; quantity and price are endogenous, and all the other variables are considered exogenous.

|  |  |
| --- | --- |
| *q* = *α*1 + *α*2*p* + *α*3*ps* + *α*4*di* + *ed* | (11.1) |
| *q* = *β*1 + *β*2*p* + *β*3*pf* + *es* | (11.2) |
| **data**("truffles", package="PoEdata")  D <- q~p+ps+di S <- q~p+pf  sys <- **list**(D,S)  instr <- **~**ps+di+pf  truff.sys <- **systemfit**(sys, inst=instr, method="2SLS", data=truffles)  **summary**(truff.sys) | | | |

##

## systemfit results

## method: 2SLS

##

## N DF SSR detRCov OLS-R2 McElroy-R2 ## system 60 53 692.47 49.803 0.43896 0.80741

##

## N DF SSR MSE RMSE R2 Adj R2 ## eq1 30 26 631.917 24.3045 4.9300 -0.02395 -0.14210 ## eq2 30 27 60.555 2.2428 1.4976 0.90188 0.89461

##

## The covariance matrix of the residuals

## eq1 eq2 ## eq1 24.3045 2.1694 ## eq2 2.1694 2.2428

##

## The correlations of the residuals

## eq1 eq2 ## eq1 1.00000 0.29384

## eq2 0.29384 1.00000

##

##

## 2SLS estimates for 'eq1' (equation 1)

## Model Formula: q ~ p + ps + di

## Instruments: ~ps + di + pf

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -4.27947 5.54388 -0.7719 0.44712

## p -0.37446 0.16475 -2.2729 0.03154 \*

## ps 1.29603 0.35519 3.6488 0.00116 \*\*

## di 5.01398 2.28356 2.1957 0.03724 \*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 4.92996 on 26 degrees of freedom

## Number of observations: 30 Degrees of Freedom: 26

## SSR: 631.91714 MSE: 24.30451 Root MSE: 4.92996

## Multiple R-Squared: -0.02395 Adjusted R-Squared: -0.1421

##

##

## 2SLS estimates for 'eq2' (equation 2)

## Model Formula: q ~ p + pf

## Instruments: ~ps + di + pf

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 20.032802 1.223115 16.378 1.554e-15 \*\*\* ## p 0.337982 0.024920 13.563 1.434e-13 \*\*\* ## pf -1.000909 0.082528 -12.128 1.946e-12 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.49759 on 27 degrees of freedom

## Number of observations: 30 Degrees of Freedom: 27

## SSR: 60.55457 MSE: 2.24276 Root MSE: 1.49759

## Multiple R-Squared: 0.90188 Adjusted R-Squared: 0.89461

The output of the systemfit() function shows the estimates by structural equation:eq1is the demand function, where, as expected, price has a negative sign, andeq2‘ is the supply equation, with a positive sign for price.

Table 11.1: Reduced form for quantity

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 7.8951 | 3.2434 | 2.4342 | 0.0221 |
| Ps | 0.6564 | 0.1425 | 4.6051 | 0.0001 |
| di | 2.1672 | 0.7005 | 3.0938 | 0.0047 |
| pf | -0.5070 | 0.1213 | -4.1809 | 0.0003 |

Table 11.2: Reduced form for price

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 7.8951 | 3.2434 | 2.4342 | 0.0221 |
| ps | 0.6564 | 0.1425 | 4.6051 | 0.0001 |
| di | 2.1672 | 0.7005 | 3.0938 | 0.0047 |
| pf | -0.5070 | 0.1213 | -4.1809 | 0.0003 |

By evaluating the reduced form equation using OLS, one can determinne the effects of changes in exogenous variables on the **equilibrium** market price and quantity, while the structural equations show the effects of such changes on the quantity demanded, respectively on the quantity supplied. Estimating the structural equations by such methods as 2SLS is, in fact, estimating the market demand and supply curves, which is extremly useful for economic analysis. Estimating the reduced forms, while being useful for prediction, does not allow for deep analysis - it only gives the equilibrium point, not the whole curves.

Q.red <- **lm**(q~ps+di+pf, data=truffles)

P.red <- **lm**(q~ps+di+pf, data=truffles) **kable**(**tidy**(Q.red), digits=4, caption="Reduced form for quantity")

**kable**(**tidy**(P.red), digits=4, caption="Reduced form for price")

Tables 11.1 and 11.2 show that all the exogenous variables have significant effects on the equilibrium quantity and price and have the expected signs.

The *fultonfish* dataset provides another demand and supply example where the simultaneous equations method can be applied. The purpose of this example is to emphasize that the exogenous variables that are key for identification must be statistically significant. Otherwise, the structural equation that needs to be identified by those variables cannot be reliably estimated. The remaining equations in the structural system are, however, not affected.

*log*(*quan*) = *α*1 + *α*2*log*(*price*) + *α*3*mon* + *α*4*tue* + *α*4*wed* + *α*5*thu* + *eD* (11.3)

*log*(*quan*) = *β*1 + *β*2*log*(*price*) + *β*3*stormy* + *eS* (11.4)

In the *fultonfish* example, the endogenous variables are *lprice*, the log of price, and *lquan*; the exogenous variables are the indicator variables for the day of the week, and whether the catching day was stormy. The identification variable for the demand equation is *stormy*, which will only show up in the supply equation; the identification variables for the supply equation will be *mon*, *tue*, *wed*, and *thu*. *log*(*q*) = *π*11 + *π*21*mon* + *π*31*tue* + *π*41*wed* + *π*51*thu* + *π*61*stormy* + *ν*1 (11.5)

*log*(*p*) = *π*12 + *π*22*mon* + *π*32*tue* + *π*42*wed* + *π*52*thu* + *π*62*stormy* + *ν*2 (11.6)

Now, let us consider the reduced form equations (Equations 11.5 and 11.6). Since the endogenous variable that appears in the right-hand side of the structural equations (Equations 11.3 and 11.4) is *price*, the *price* reduced equation (Equation 11.6) is essential for evaluating the identification state of the model. Let us focus on this equation. If the weekday indicators are all insignificant, the supply equation cannot be identified; if *stormy* turns out insignificant, the demand equation cannot be identified; if the weekday indicators are insignificat but *stormy* is significant the supply is not identified, but the demand is; if at least one weekday indicator turns out significant but *stormy* turns out insignificant, the demand equation is not identified but the supply equation is. Equations 11.3 and 11.4 display the structural demand and supply equations for the *fultonfish* example.

**data**("fultonfish", package="PoEdata")

fishQ.ols <- **lm**(lquan~mon+tue+wed+thu+stormy, data=fultonfish)

**kable**(**tidy**(fishQ.ols), digits=4, caption="Reduced 'Q' equation for the fultonfish example")

fishP.ols <- **lm**(lprice~mon+tue+wed+thu+stormy, data=fultonfish)

**kable**(**tidy**(fishP.ols), digits=4, caption="Reduced 'P' equation for the fultonfish example")

The relevant equation for evaluating identification is shown in Table 11.4, which is the price reduced equation. The results show that the weekday indicators are not Table 11.3: Reduced ’Q’ equation for the fultonfish example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 8.8101 | 0.1470 | 59.9225 | 0.0000 |
| mon | 0.1010 | 0.2065 | 0.4891 | 0.6258 |
| tue | -0.4847 | 0.2011 | -2.4097 | 0.0177 |
| wed | -0.5531 | 0.2058 | -2.6875 | 0.0084 |
| thu | 0.0537 | 0.2010 | 0.2671 | 0.7899 |
| stormy | -0.3878 | 0.1437 | -2.6979 | 0.0081 |

Table 11.4: Reduced ’P’ equation for the fultonfish example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -0.2717 | 0.0764 | -3.5569 | 0.0006 |
| mon | -0.1129 | 0.1073 | -1.0525 | 0.2950 |
| tue | -0.0411 | 0.1045 | -0.3937 | 0.6946 |
| wed | -0.0118 | 0.1069 | -0.1106 | 0.9122 |
| thu | 0.0496 | 0.1045 | 0.4753 | 0.6356 |
| stormy | 0.3464 | 0.0747 | 4.6387 | 0.0000 |

significant, which will make the 2SLS estimation of the supply equation unreliable; the coefficient on *stormy* is significant, thus the estimation of the (structural) demand equation will be reliable. The following code sequence and output show the 2SLS estimates of the demand and supply (the structural) equations.

|  |
| --- |
| fish.D <- lquan~lprice+mon+tue+wed+thu fish.S <- lquan~lprice+stormy fish.eqs <- **list**(fish.D, fish.S) fish.ivs <- **~**mon+tue+wed+thu+stormy fish.sys <- **systemfit**(fish.eqs, method="2SLS",  inst=fish.ivs, data=fultonfish)  **summary**(fish.sys) |

##

## systemfit results

## method: 2SLS

##

## N DF SSR detRCov OLS-R2 McElroy-R2

## system 222 213 109.61 0.1073 0.09424 -0.59781

##

## N DF SSR MSE RMSE R2 Adj R2 ## eq1 111 105 52.090 0.49610 0.70434 0.13912 0.09813 ## eq2 111 108 57.522 0.53261 0.72980 0.04936 0.03176

##

## The covariance matrix of the residuals

## eq1 eq2 ## eq1 0.49610 0.39614

## eq2 0.39614 0.53261

##

## The correlations of the residuals

## eq1 eq2 ## eq1 1.00000 0.77065

## eq2 0.77065 1.00000

##

##

## 2SLS estimates for 'eq1' (equation 1)

## Model Formula: lquan ~ lprice + mon + tue + wed + thu

## Instruments: ~mon + tue + wed + thu + stormy

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 8.505911 0.166167 51.1890 < 2.2e-16 \*\*\*

## lprice -1.119417 0.428645 -2.6115 0.010333 \*

## mon -0.025402 0.214774 -0.1183 0.906077

## tue -0.530769 0.208000 -2.5518 0.012157 \*

## wed -0.566351 0.212755 -2.6620 0.008989 \*\*

## thu 0.109267 0.208787 0.5233 0.601837

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.70434 on 105 degrees of freedom

## Number of observations: 111 Degrees of Freedom: 105

## SSR: 52.09032 MSE: 0.4961 Root MSE: 0.70434

## Multiple R-Squared: 0.13912 Adjusted R-Squared: 0.09813

##

##

## 2SLS estimates for 'eq2' (equation 2)

## Model Formula: lquan ~ lprice + stormy

## Instruments: ~mon + tue + wed + thu + stormy

##

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 8.6283544 0.3889702 22.1826 <2e-16 \*\*\*

## lprice 0.0010593 1.3095470 0.0008 0.9994 ## stormy -0.3632461 0.4649125 -0.7813 0.4363

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.7298 on 108 degrees of freedom

## Number of observations: 111 Degrees of Freedom: 108

## SSR: 57.52184 MSE: 0.53261 Root MSE: 0.7298

## Multiple R-Squared: 0.04936 Adjusted R-Squared: 0.03176 In the output of the 2SLS estimation, eq1 is the demand equation, and eq2 is the supply. As we have seen the demand equation is identified, i.e., reliable, while the supply equation is not. A solution might be to find better instruments, other than the weekdays for the demand equation. Finding valid instruments is, however, a difficult task in many problems.

Lab 2 Time Series: Non-stationarity

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!*  **library**(tseries) *# for ADF unit root tests* **library**(dynlm)  **library**(nlWaldTest) *# for the `nlWaldtest()` function* **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 datasets* **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich) **library**(knitr) *#for kable()* **library**(forecast) |

New package: tseries (Trapletti and Hornik 2016).

A time series is nonstationary if its distribution, in particular its mean, variance, or timewise covariance change over time. Nonstationary time series cannot be used in regression models because they may create **spurious regression**, a false relationship due to, for instance, a common trend in otherwise unrelated variables. Two or more nonstationary series can still be part of a regression model if they are **cointegrated**, that is, they are in a stationary relationship of some sort.

We are concerned with testing time series for nonstationarity and finding out how can we transform nonstationary time series such that we can still use them in regression analysis.

**data**("usa", package="PoEdata")

usa.ts <- **ts**(usa, start=**c**(1984,1), end=**c**(2009,4), frequency=4)

Dgdp <- **diff**(usa.ts[,1])

177

Table 12.1: Time series data frame constructed with ’ts.union’

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| gdp | inf | f | b | Dgdp | Dinf | Df | Db |
| 3807.4 | 9.47 | 9.69 | 11.19 | NA | NA | NA | NA |
| 3906.3 | 10.03 | 10.56 | 12.64 | 98.9 | 0.56 | 0.87 | 1.45 |
| 3976.0 | 10.83 | 11.39 | 12.64 | 69.7 | 0.80 | 0.83 | 0.00 |
| 4034.0 | 11.51 | 9.27 | 11.10 | 58.0 | 0.68 | -2.12 | -1.54 |
| 4117.2 | 10.51 | 8.48 | 10.68 | 83.2 | -1.00 | -0.79 | -0.42 |
| 4175.7 | 9.24 | 7.92 | 9.76 | 58.5 | -1.27 | -0.56 | -0.92 |
| Dinf <- **diff**(usa.ts[,"inf"]) Df <- **diff**(usa.ts[,"f"]) Db <- **diff**(usa.ts[,"b"])  usa.ts.df <- **ts.union**(gdp=usa.ts[,1], *# package tseries*  inf=usa.ts[,2], f=usa.ts[,3], b=usa.ts[,4],  Dgdp,Dinf,Df,Db, dframe=TRUE)  **plot**(usa.ts.df$gdp) **plot**(usa.ts.df$Dgdp) **plot**(usa.ts.df$inf) **plot**(usa.ts.df$Dinf) **plot**(usa.ts.df$f) **plot**(usa.ts.df$Df) **plot**(usa.ts.df$b) **plot**(usa.ts.df$Db) | | | | | | | | | |

A novelty in the above code sequence is the use of the function ts.union, wich binds together several time series, with the possibility of constructing a data frame. Table

12.1 presents the head of this data frame.

**kable**(**head**(usa.ts.df),

caption="Time series data frame constructed with 'ts.union'")

## 12.1 AR(1), the First-Order Autoregressive Model

An AR(1) **stochastic process** is defined by Equation 12.1, where the error term is sometimes called “innovation” or “shock.”

*12.1. AR(1), THE FIRST-ORDER AUTOREGRESSIVE MODEL*

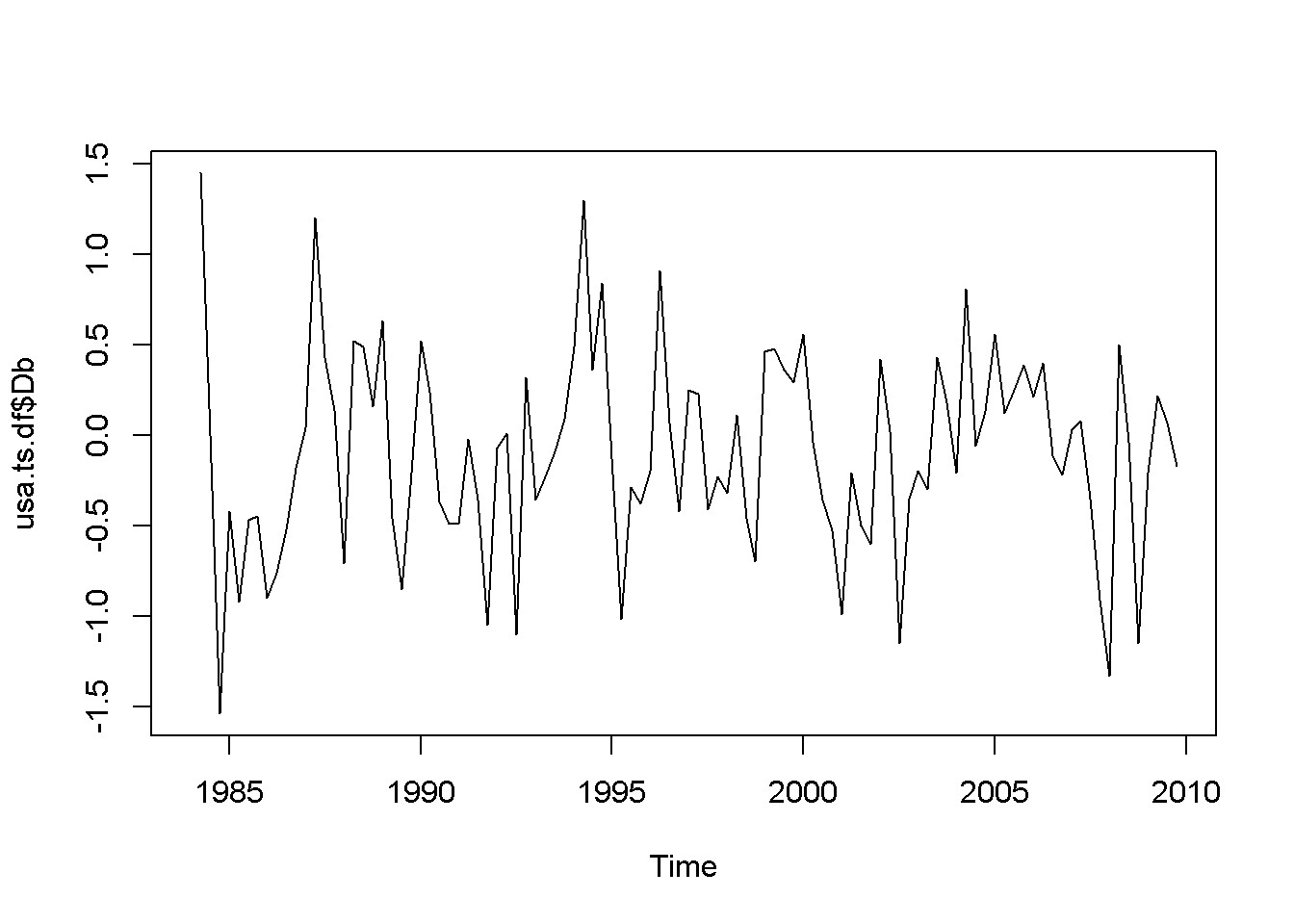
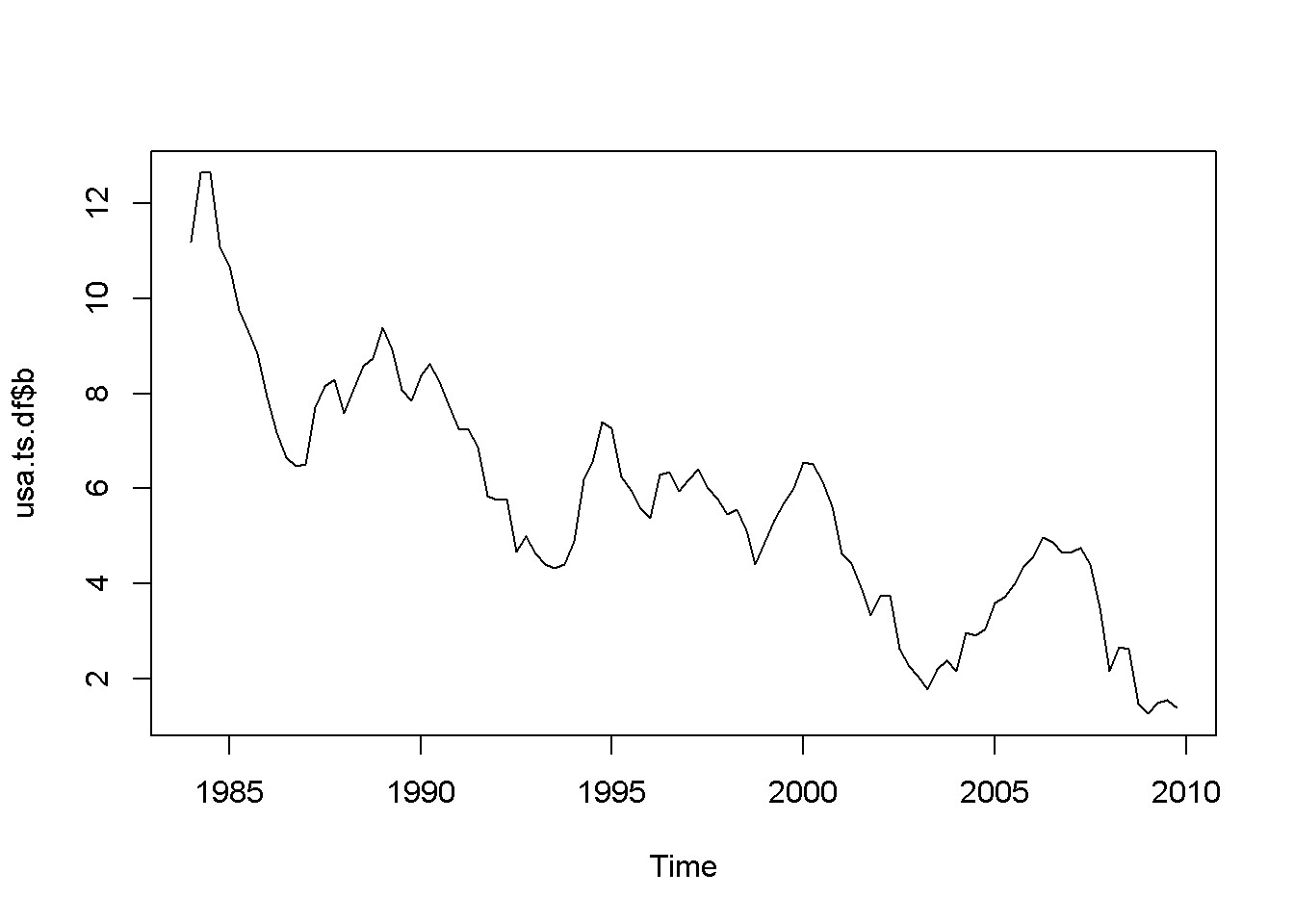
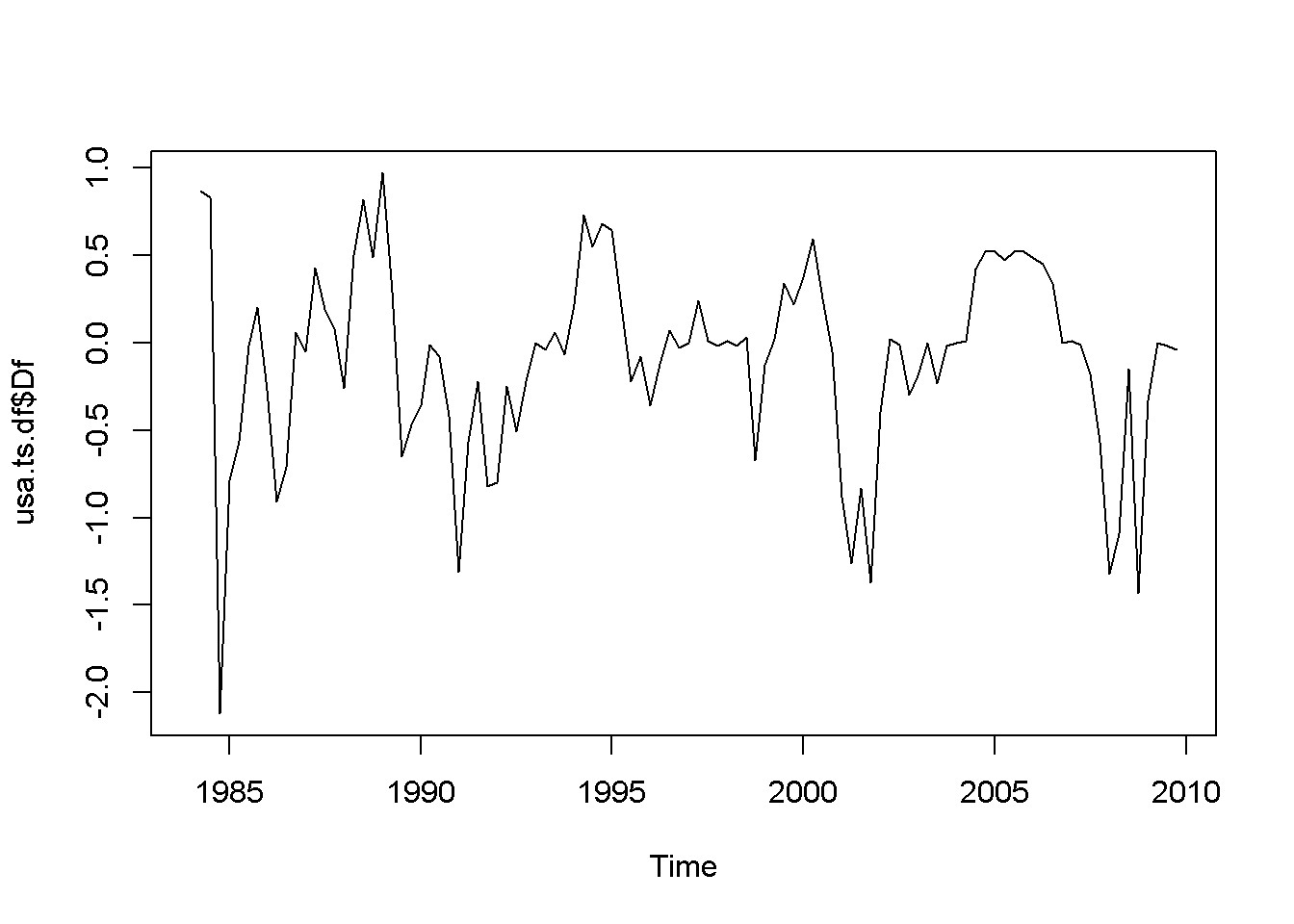
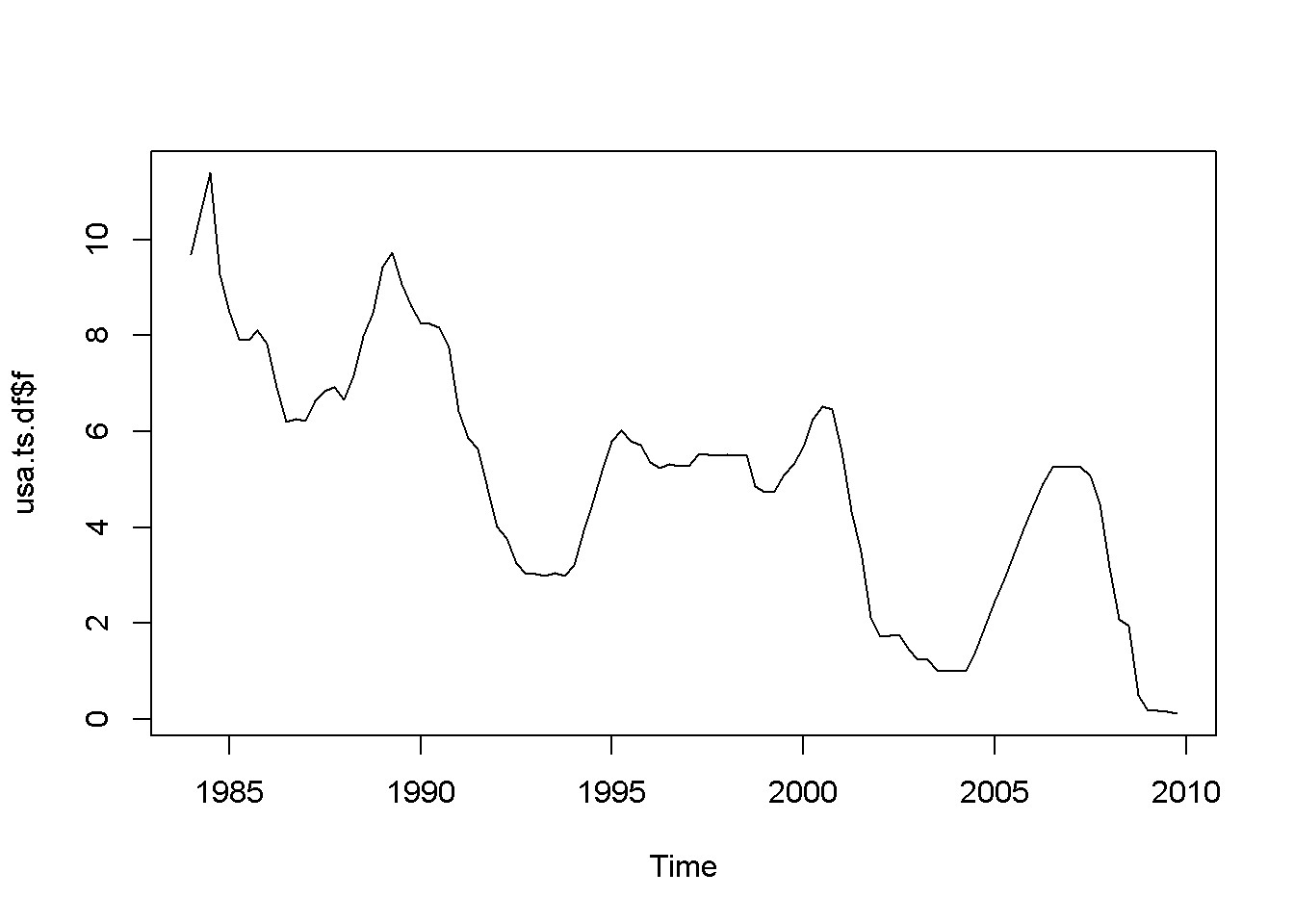
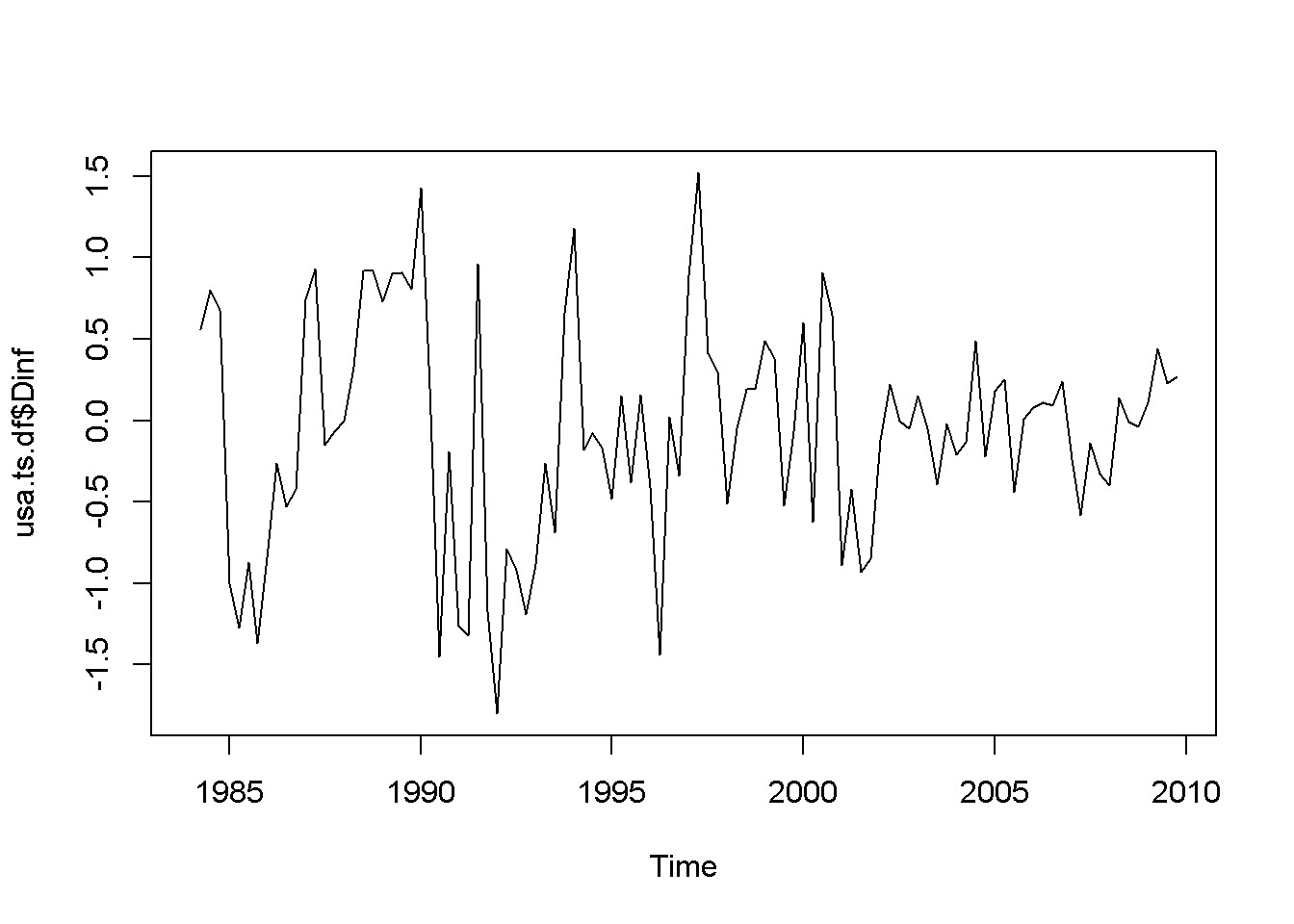
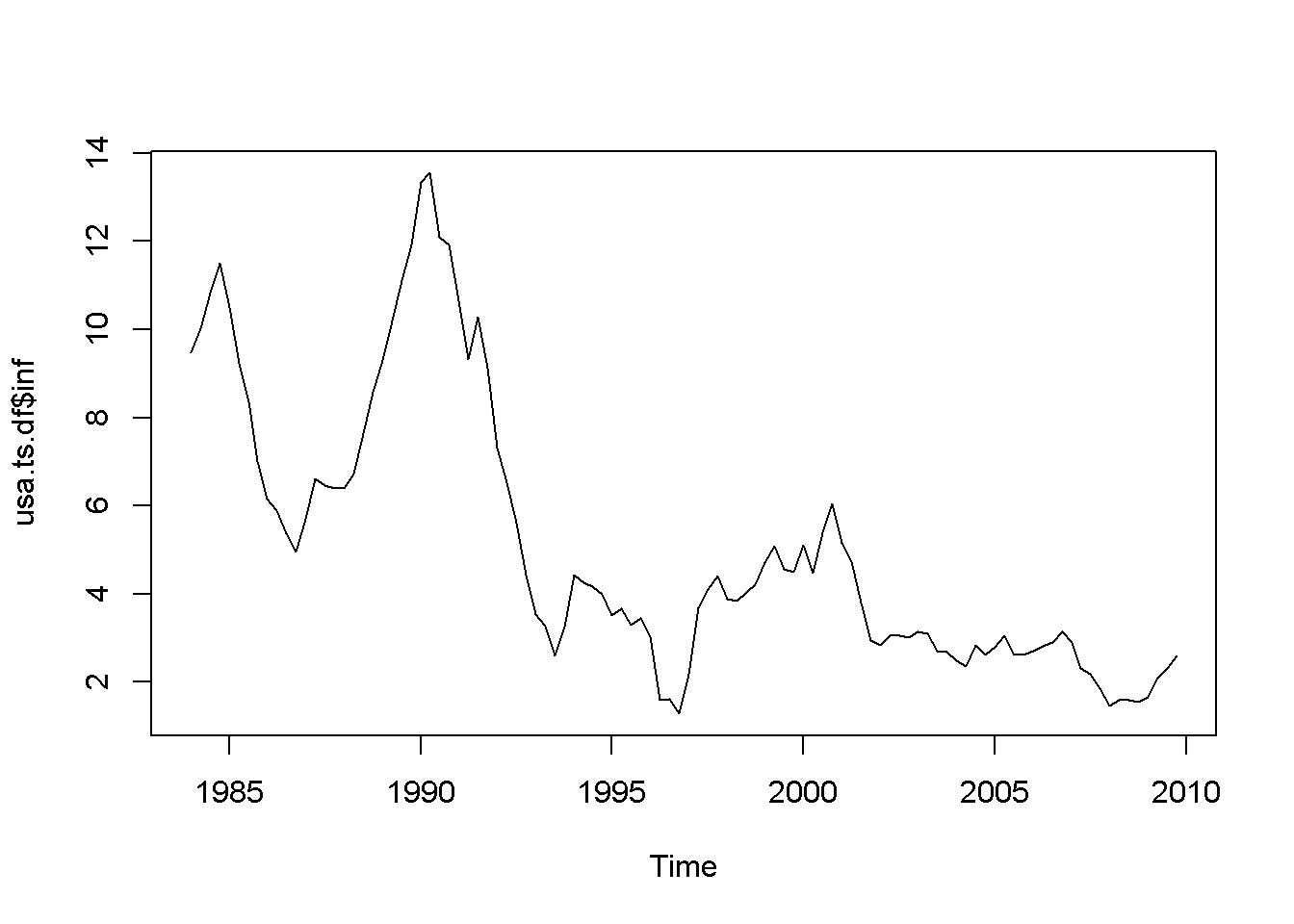
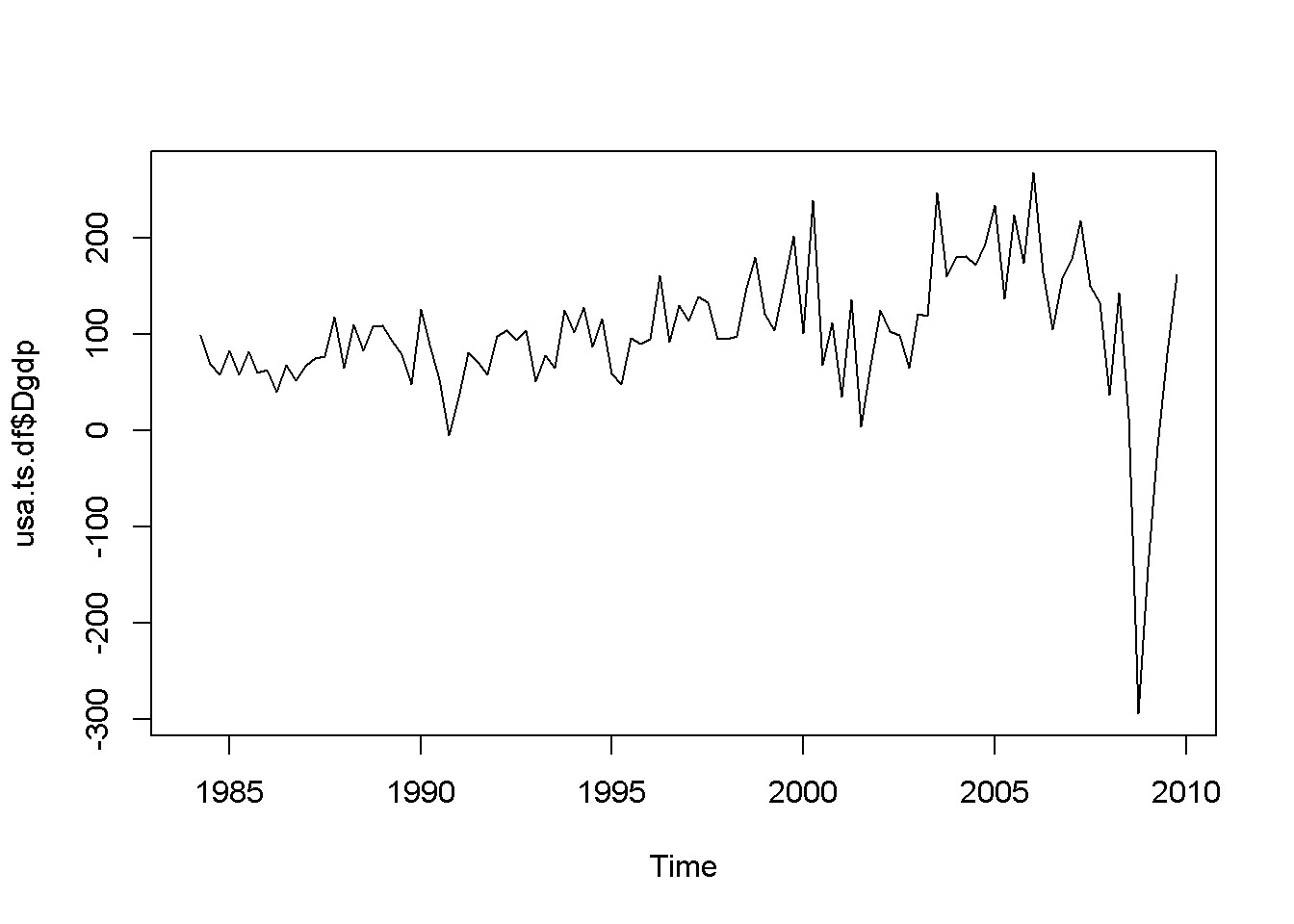
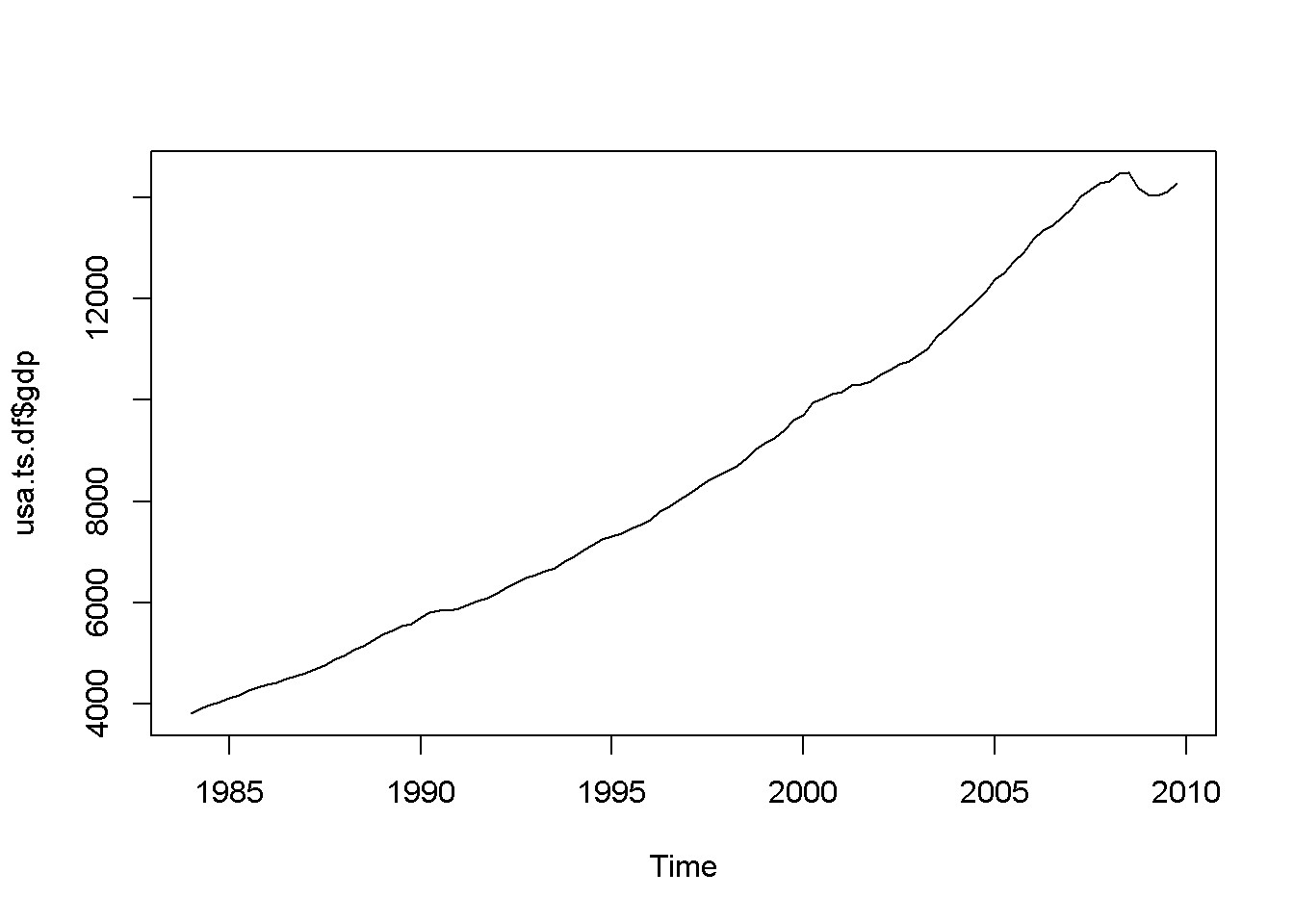


Figure 12.1: Various time series to illustrate nonstationarity

*yt* = *ρyt−*1 + *νt, |ρ| <* 1 (12.1)

The AR(1) process is stationary if *|ρ| <* 1; when *ρ* = 1, the process is called **random walk**. The next code piece plots various AR(1) processes, with or without a constant, with or without trend (time as a term in the random process equation), with *ρ* lesss or equal to 1. The generic equation used to draw the diagrams is given in Equation

12.2.

*yt* = *α* + *λt* + *ρyt−*1 + *νt* (12.2)

|  |
| --- |
| N <- 500 a <- 1 l <- 0.01 rho <- 0.7  **set.seed**(246810) v <- **ts**(**rnorm**(N,0,1))  y <- **ts**(**rep**(0,N)) for (t in 2:N){ y[t]<- rho\*y[t-1]+v[t]  }  **plot**(y,type='l', ylab="rho\*y[t-1]+v[t]") **abline**(h=0)  y <- **ts**(**rep**(0,N)) for (t in 2:N){ y[t]<- a+rho\*y[t-1]+v[t]  }  **plot**(y,type='l', ylab="a+rho\*y[t-1]+v[t]") **abline**(h=0)  y <- **ts**(**rep**(0,N)) for (t in 2:N){  y[t]<- a+l\***time**(y)[t]+rho\*y[t-1]+v[t]  }  **plot**(y,type='l', ylab="a+l\*time(y)[t]+rho\*y[t-1]+v[t]") **abline**(h=0)  y <- **ts**(**rep**(0,N)) for (t in 2:N){ |

*12.2. SPURIOUS REGRESSION*

|  |
| --- |
| y[t]<- y[t-1]+v[t]  }  **plot**(y,type='l', ylab="y[t-1]+v[t]") **abline**(h=0) a <- 0.1 y <- **ts**(**rep**(0,N)) for (t in 2:N){ y[t]<- a+y[t-1]+v[t]  }  **plot**(y,type='l', ylab="a+y[t-1]+v[t]") **abline**(h=0)  y <- **ts**(**rep**(0,N)) for (t in 2:N){  y[t]<- a+l\***time**(y)[t]+y[t-1]+v[t]  }  **plot**(y,type='l', ylab="a+l\*time(y)[t]+y[t-1]+v[t]") **abline**(h=0) |

## 12.2 Spurious Regression

Nonstationarity can lead to **spurious regression**, an apparent relationship between variables that are, in reality not related. The following code sequence generates two independent random walk processes, *y* and *x*, and regresses *y* on *x*.

T <- 1000 **set.seed**(1357) y <- **ts**(**rep**(0,T)) vy <- **ts**(**rnorm**(T)) for (t in 2:T){

y[t] <- y[t-1]+vy[t]

}

**set.seed**(4365) x <- **ts**(**rep**(0,T)) vx <- **ts**(**rnorm**(T)) for (t in 2:T){

x[t] <- x[t-1]+vx[t]

}

y <- **ts**(y[300:1000])

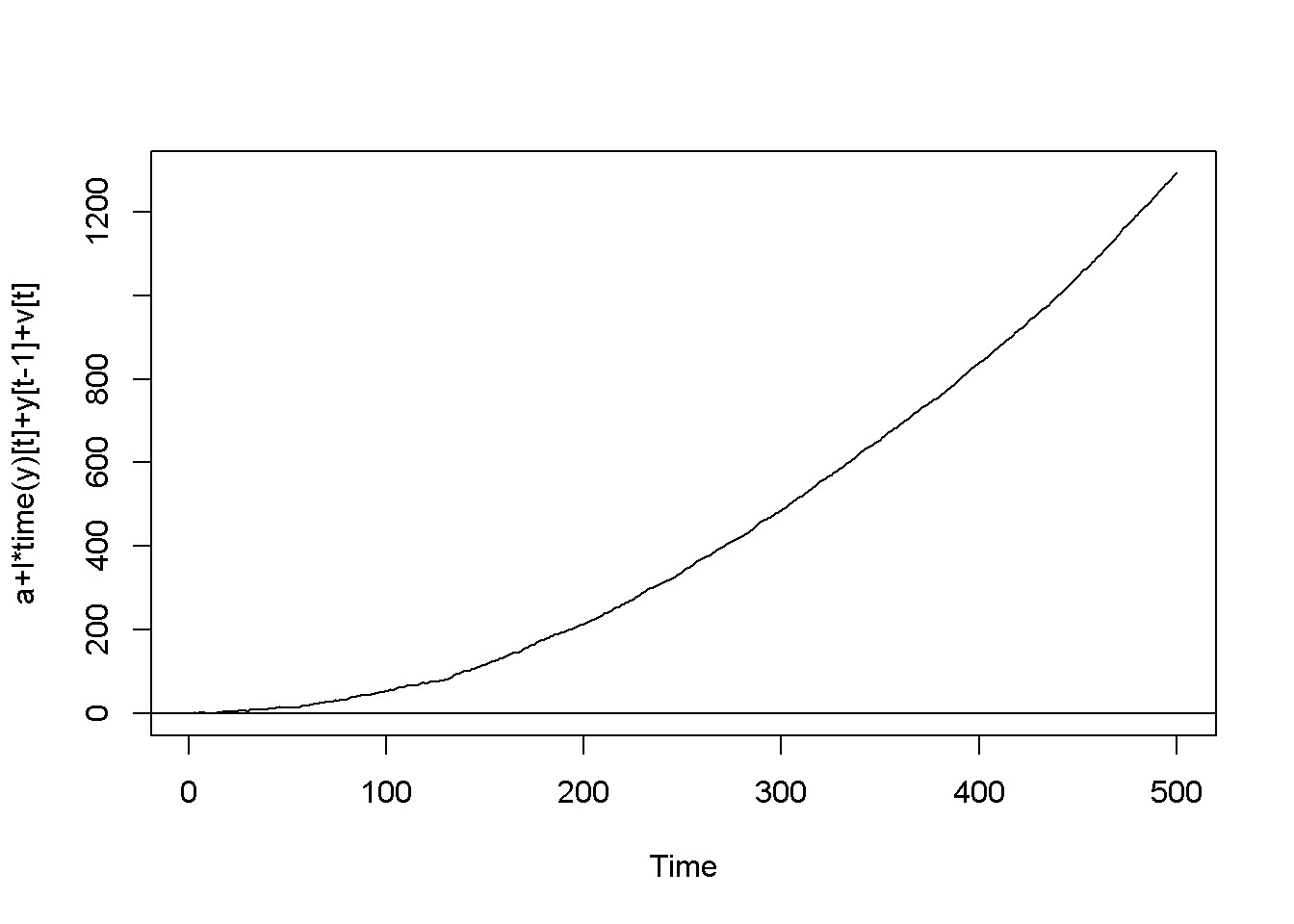
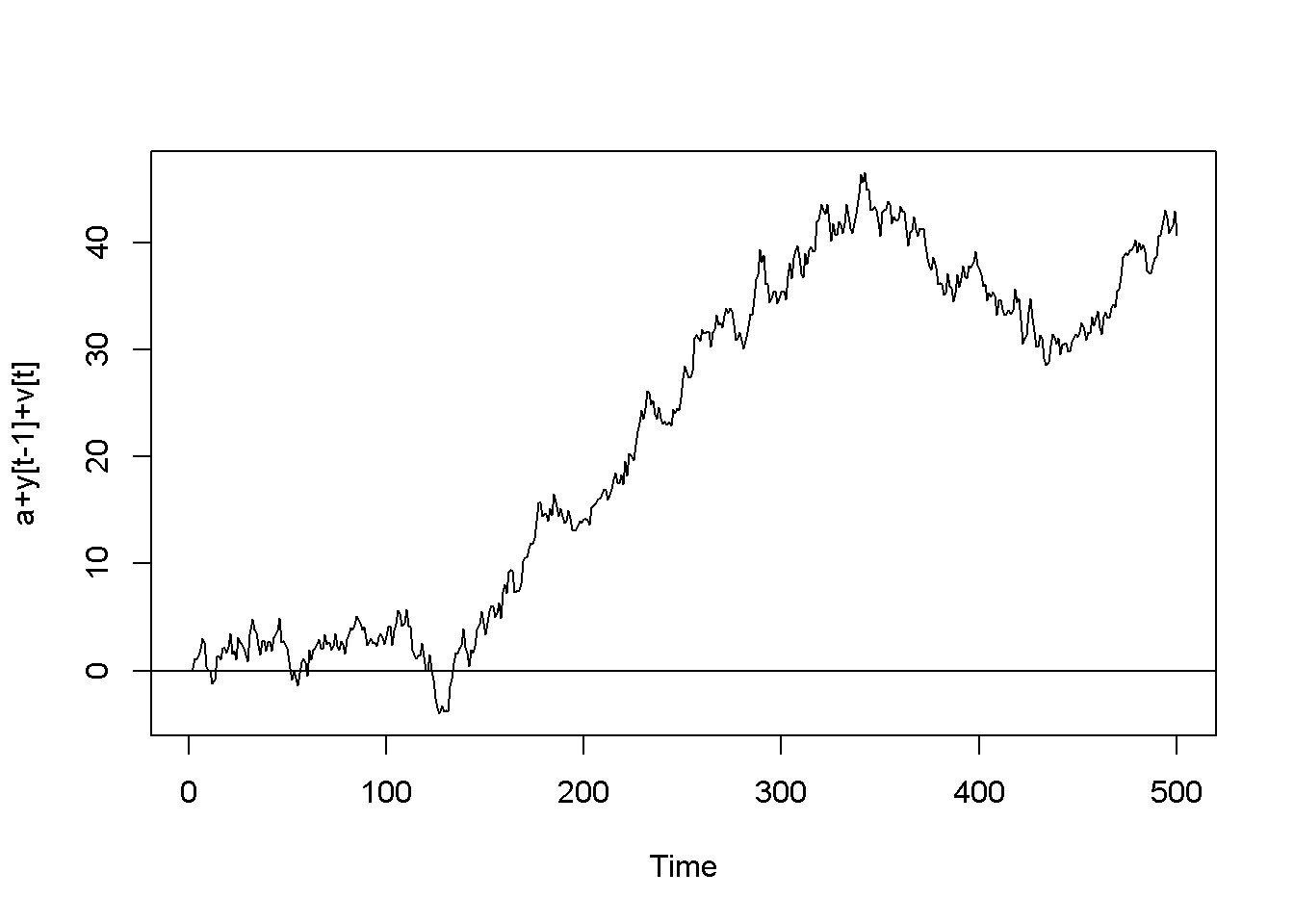
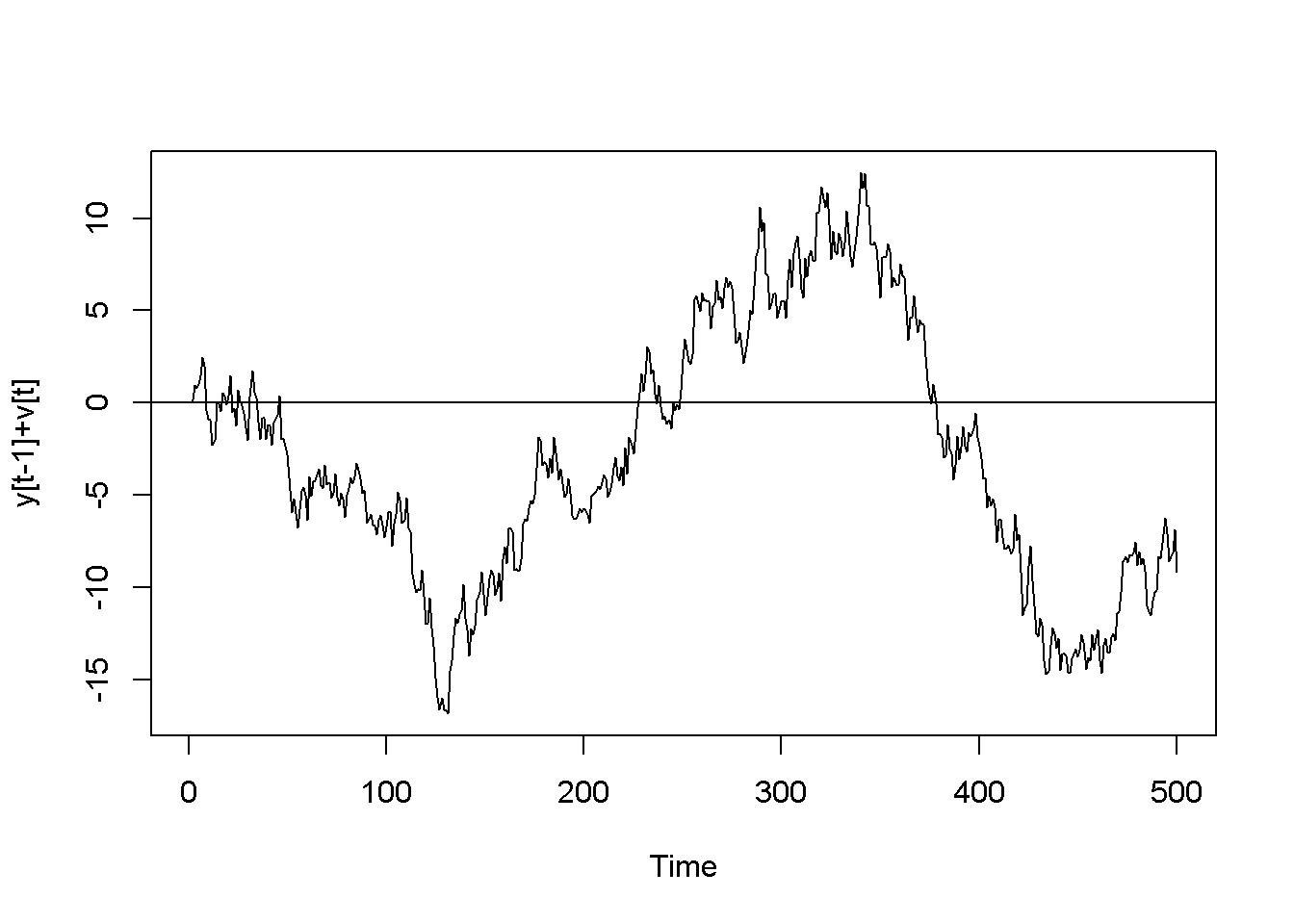
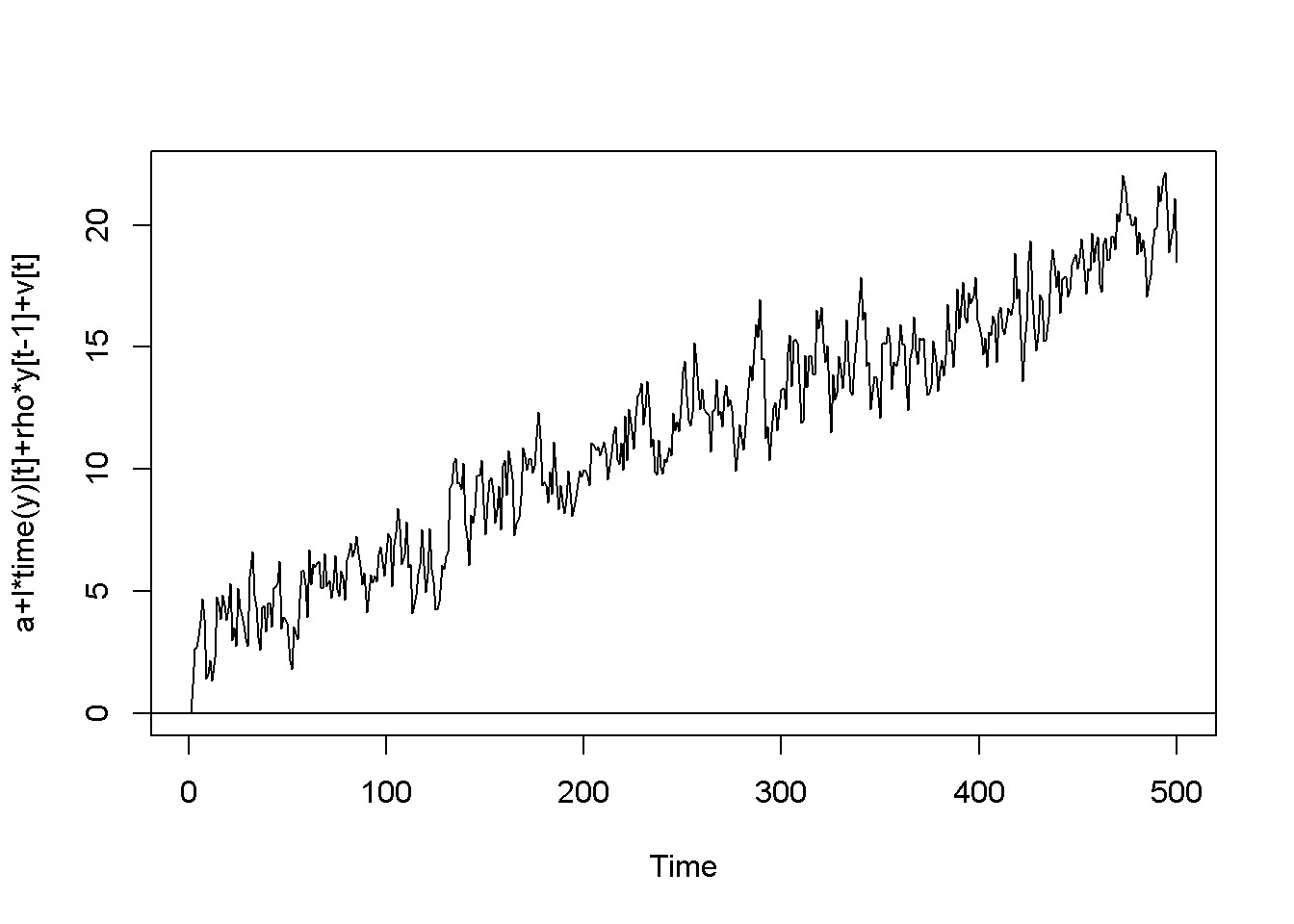
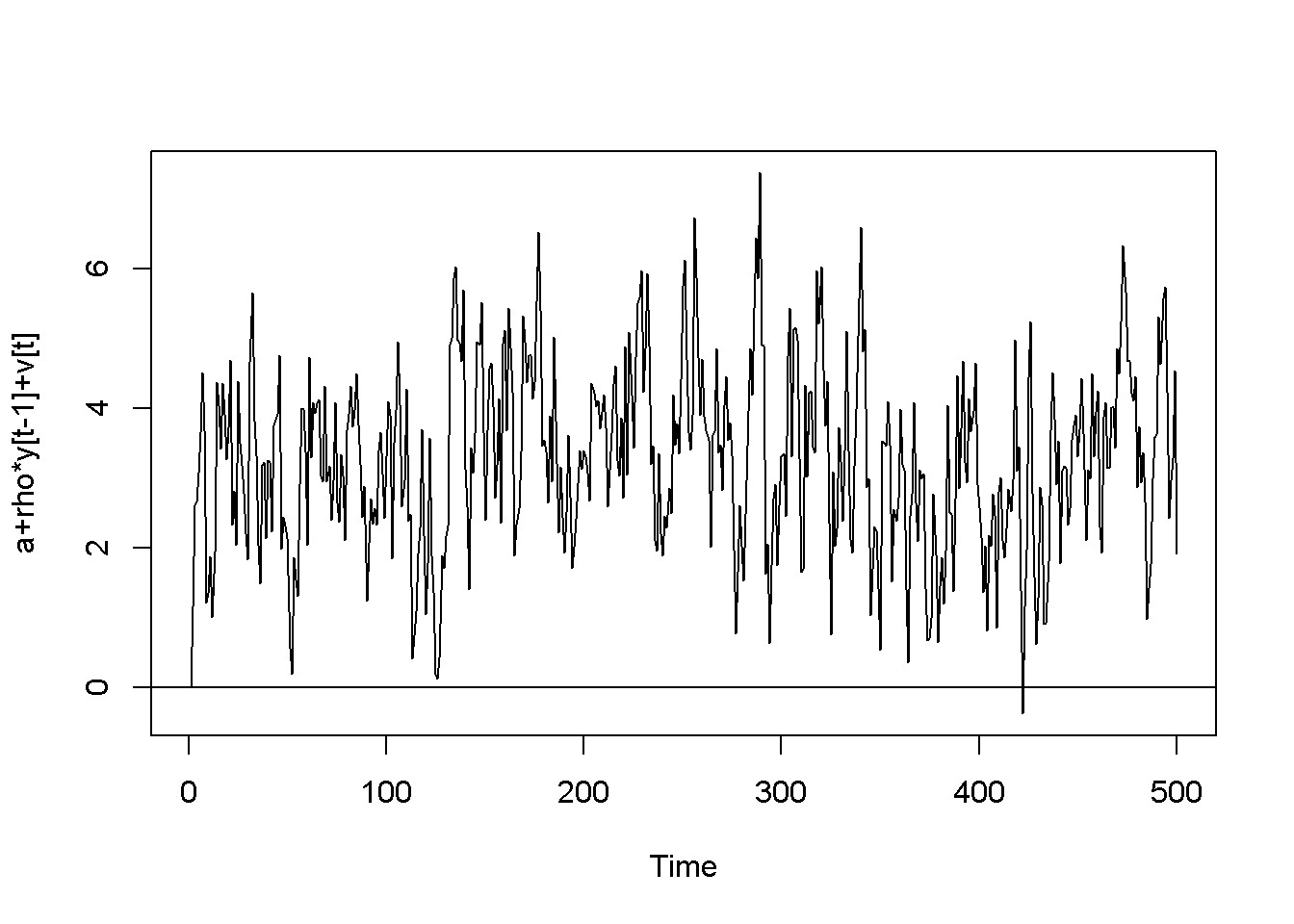
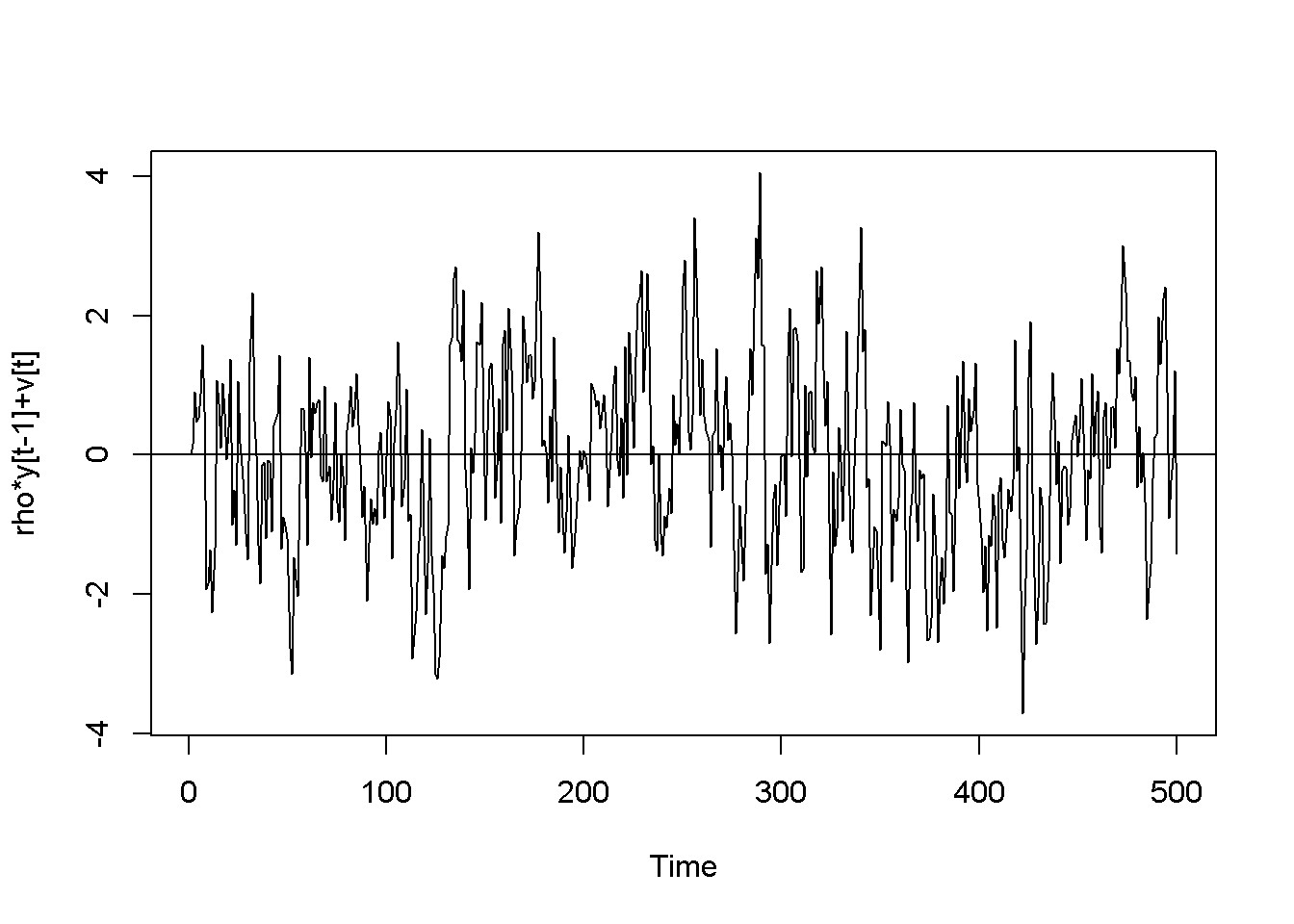


Figure 12.2: Artificially generated AR(1) processes with rho=0.7

*12.2. SPURIOUS REGRESSION*

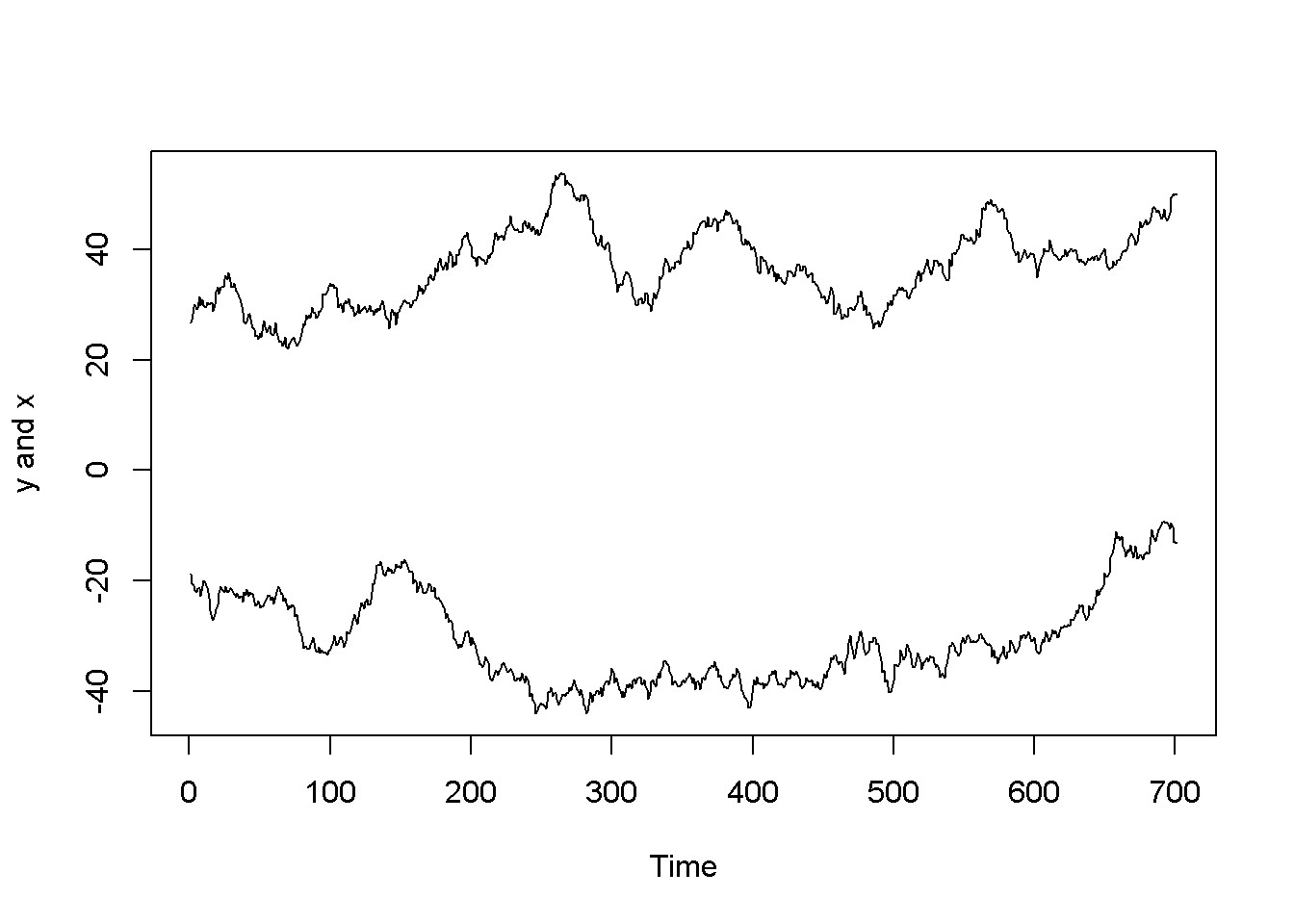


Figure 12.3: Artificially generated independent random variables

x <- **ts**(x[300:1000]) **ts.plot**(y,x, ylab="y and x")

spurious.ols <- **lm**(y~x) **summary**(spurious.ols)

##

## Call:

## lm(formula = y ~ x)

##

## Residuals:

## Min 1Q Median 3Q Max

## -12.55 -5.97 -2.45 4.51 24.68

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) -20.3871 1.6196 -12.59 < 2e-16 \*\*\*

## x -0.2819 0.0433 -6.51 1.5e-10 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 7.95 on 699 degrees of freedom

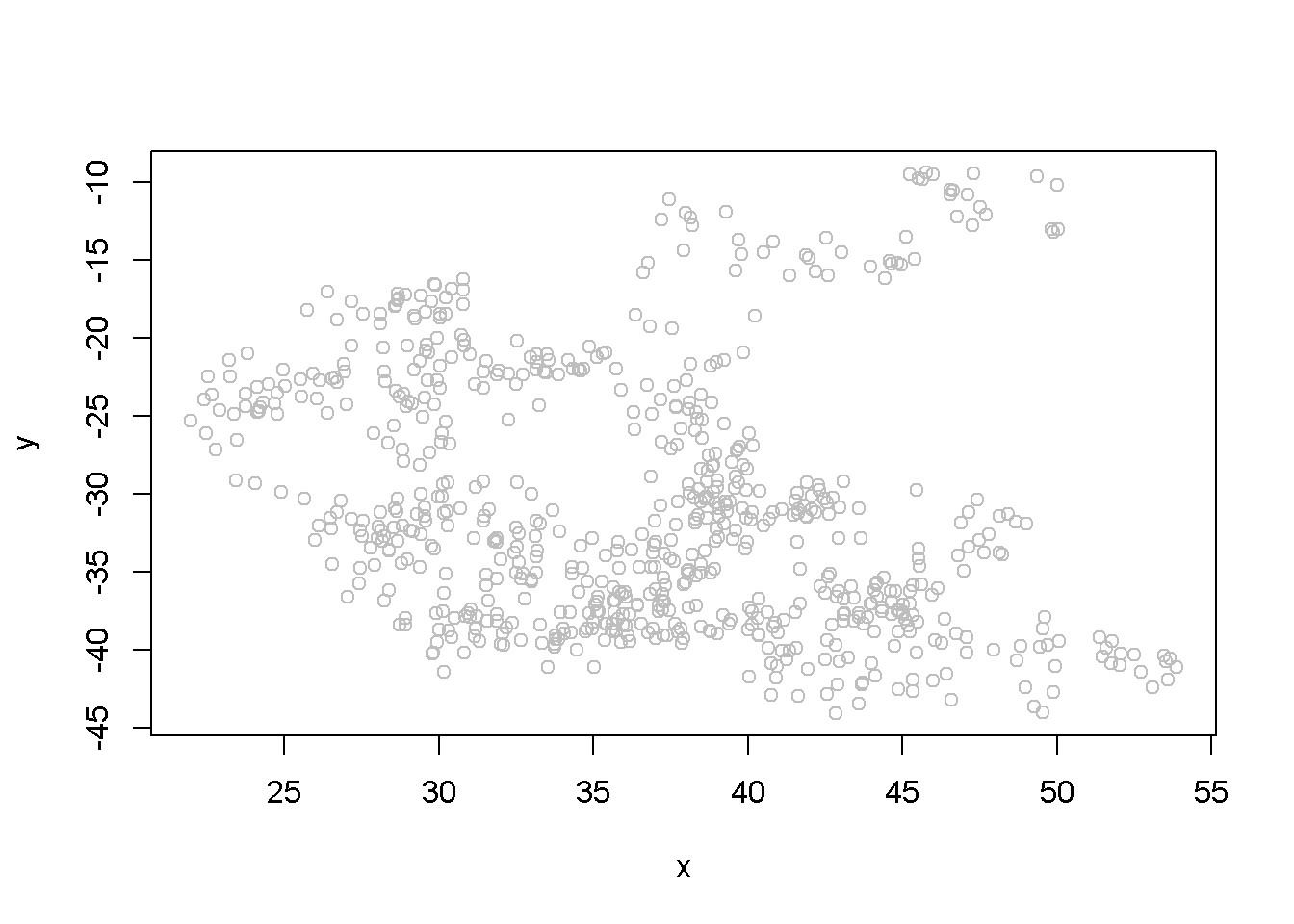


Figure 12.4: Scatter plot of artificial series y and x

## Multiple R-squared: 0.0571, Adjusted R-squared: 0.0558

## F-statistic: 42.4 on 1 and 699 DF, p-value: 1.45e-10

The summary output of the regression shows a strong correlation between the two variables, thugh they have been generated independently. (Not any two randomly generated processes need to create spurious regression, though.) Figure 12.3 depicts the two time series, *y* and *x*, and Figure 12.4 shows them in a scatterplot.

**plot**(x, y, type="p", col="grey")

Lab 3 - Unit Root Tests for Stationarity

The Dickey-Fuller test for stationarity is based on an AR(1) process as defined in Equation 12.1; if our time series seems to display a constant and trend, the basic equation is the one in Equation 12.2. According to the Dickey-Fuller test, a time series is nonstationary when *ρ* = 1, which makes the AR(1) process a random walk. The null and alternative hypotheses of the test is given in Equation 12.3.

*H*0 : *ρ* = 1*, HA* : *ρ <* 1 (12.3)

The basic AR(1) equations mentioned above are transformed, for the purpose of the

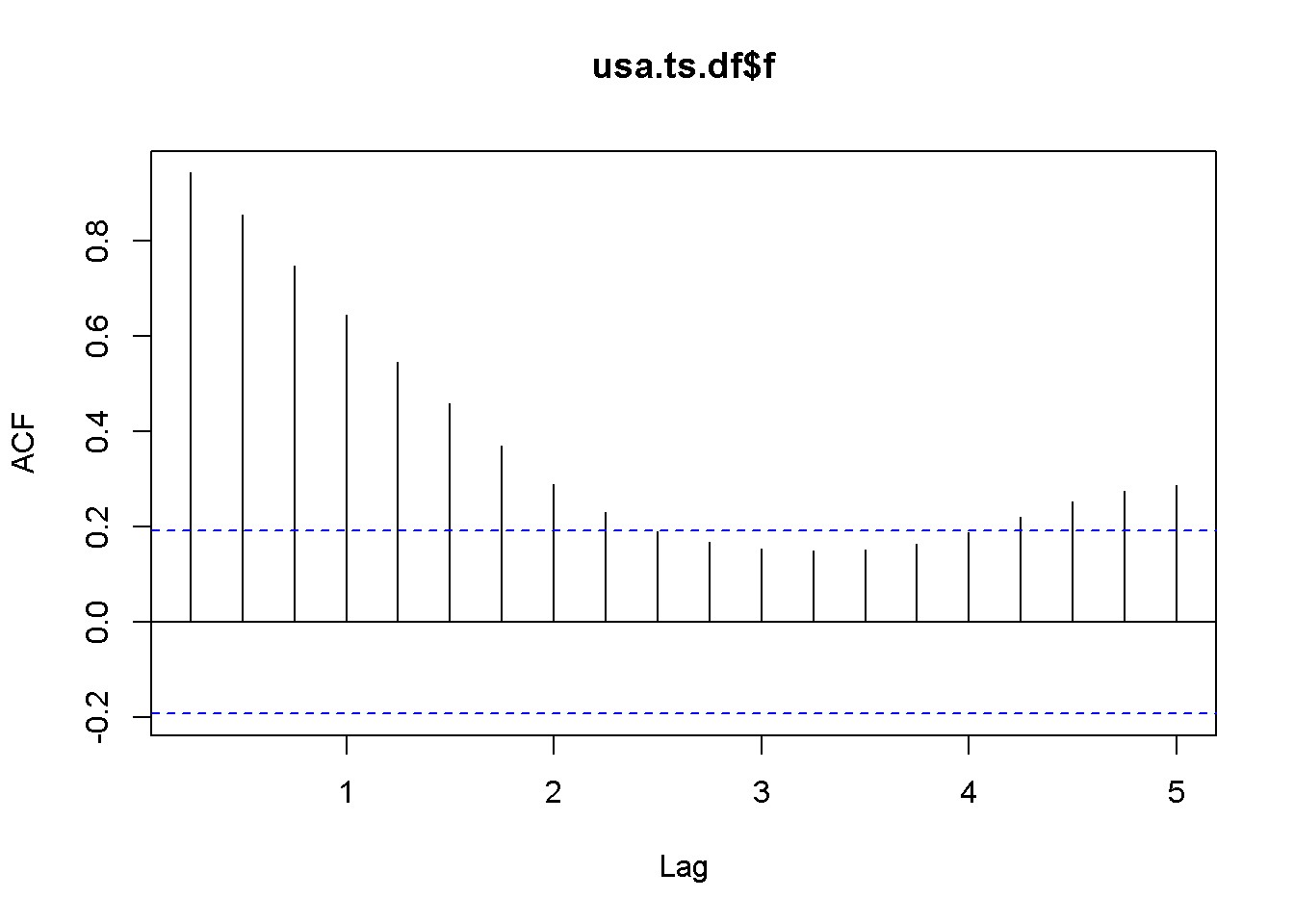
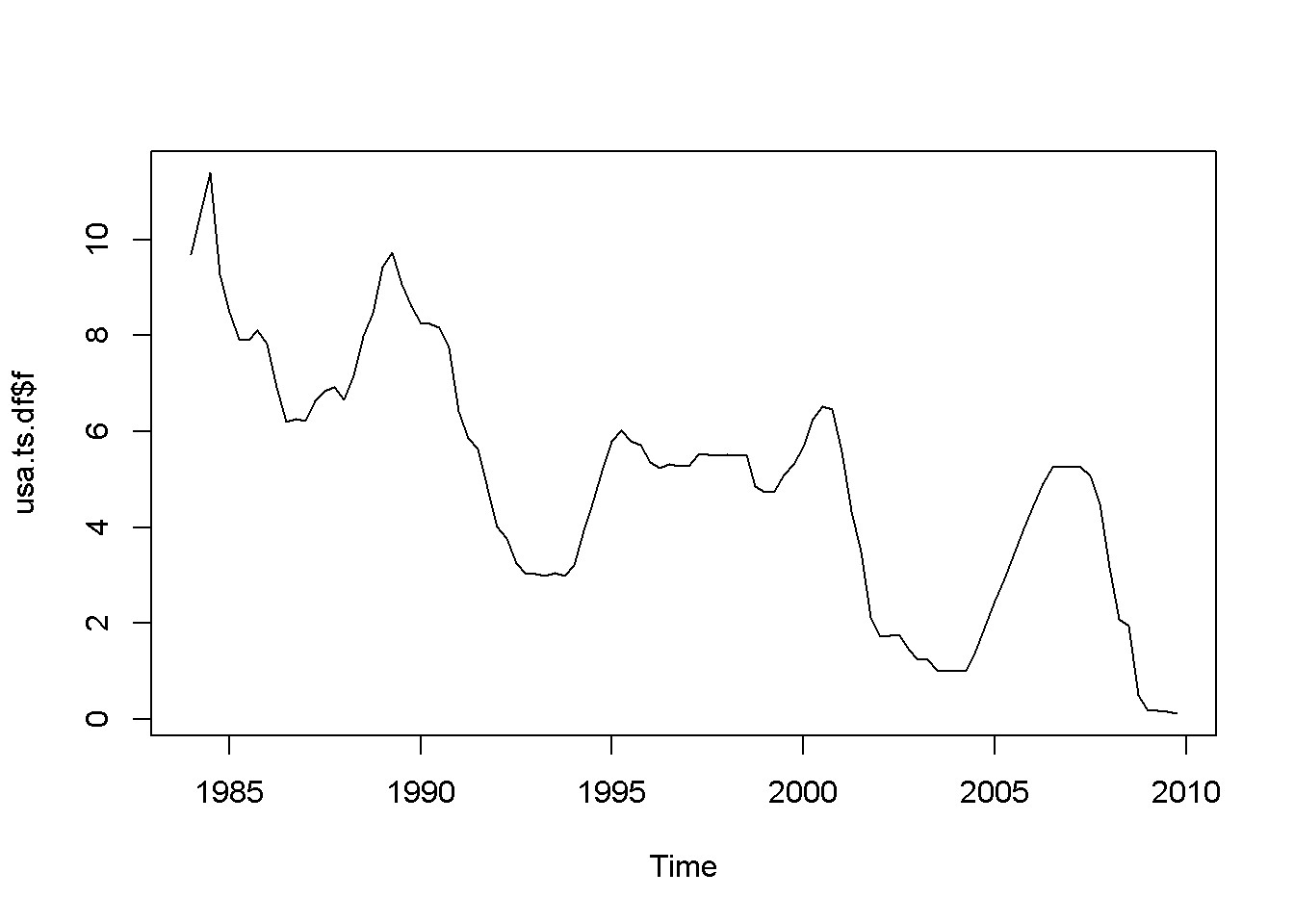


Figure 12.5: A plot and correlogram for series f in dataset usa

DF test into Equation 12.4, with the transformed hypothesis shown in Equation

12.5. **Rejecting the DF null hypothesis implies that our time series is stationary**.

∆*yt* = *α* + *γyt−*1 + *λt* + *νt* (12.4)

*H*0 : *γ* = 0*, HA* : *γ <* 0 (12.5)

An augmented DF test includes several lags of the variable tested; the number of lags to include can be assessed by examining the correlogram of the variable. The DF test can be of three types: with no constant and no trend, with constsnt and no trend, and, finally, with constant and trend. It is important to specify which DF test we want because the critical values are different for the three different types of the test. One decides which test to perform by examining a time series plot of the variable and determine if an imaginary regression line would have an intercept and a slope.

Let us apply the DF test to the *f* series in the *usa* dataset.

**plot**(usa.ts.df$f) **Acf**(usa.ts.df$f)

The time series plot in Figure 12.5 indicates both intercept and trend for our series, while the correlogram suggests including 10 lags in the DF test equation. Suppose we choose *α* = 0*.*05 for the DF test. The adf.test function does not require specifying whether the test should be conducted with constant or trend, and if no value for the number of lags is given (the argument for the number of lags is k), *R* will calculate a value for it. I would recommend always taking a look at the series’ plot and correlogram.

**adf.test**(usa.ts.df$f, k=10)

##

## Augmented Dickey-Fuller Test

##

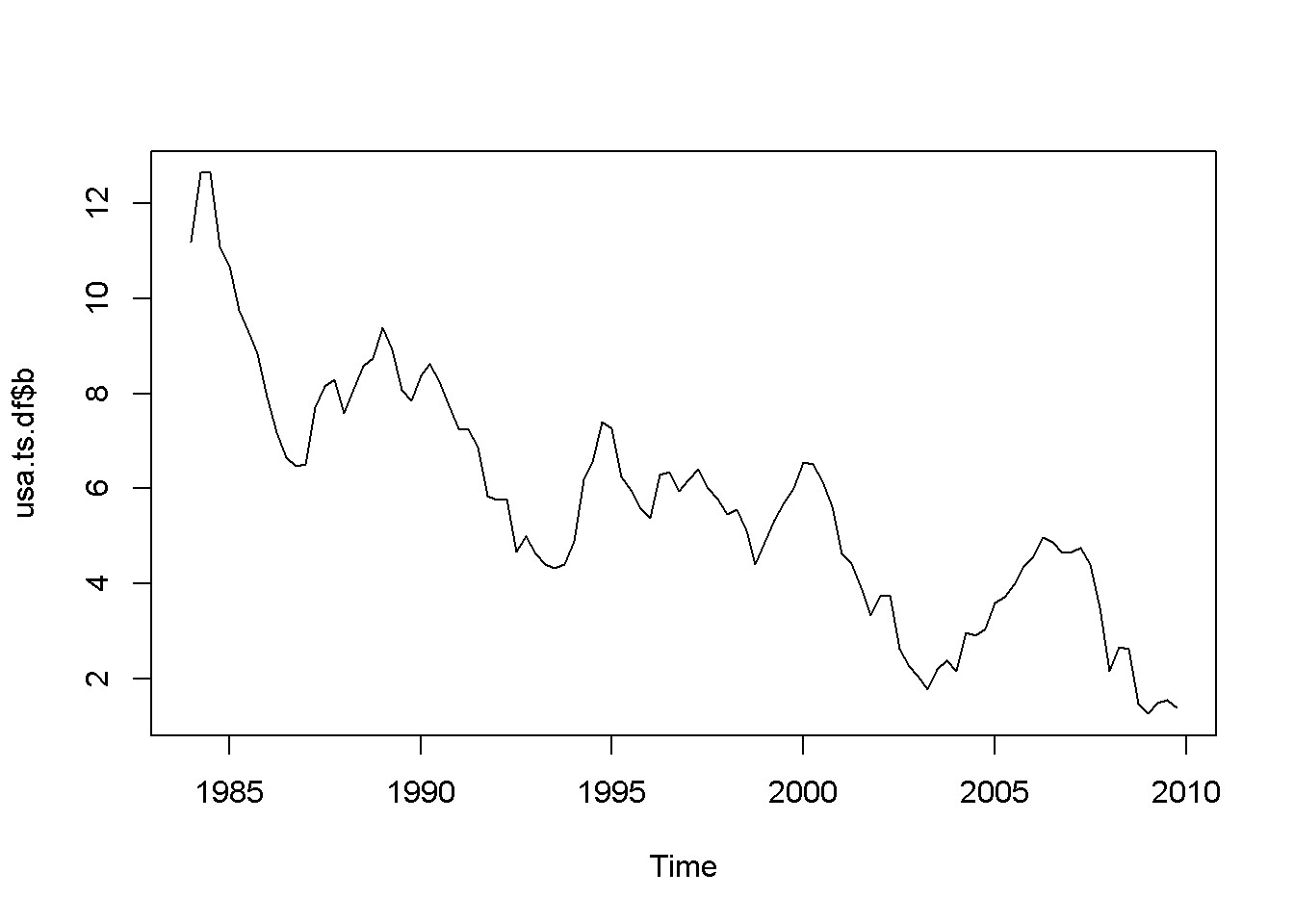
## data: usa.ts.df$f

## Dickey-Fuller = -3.373, Lag order = 10, p-value = 0.0628

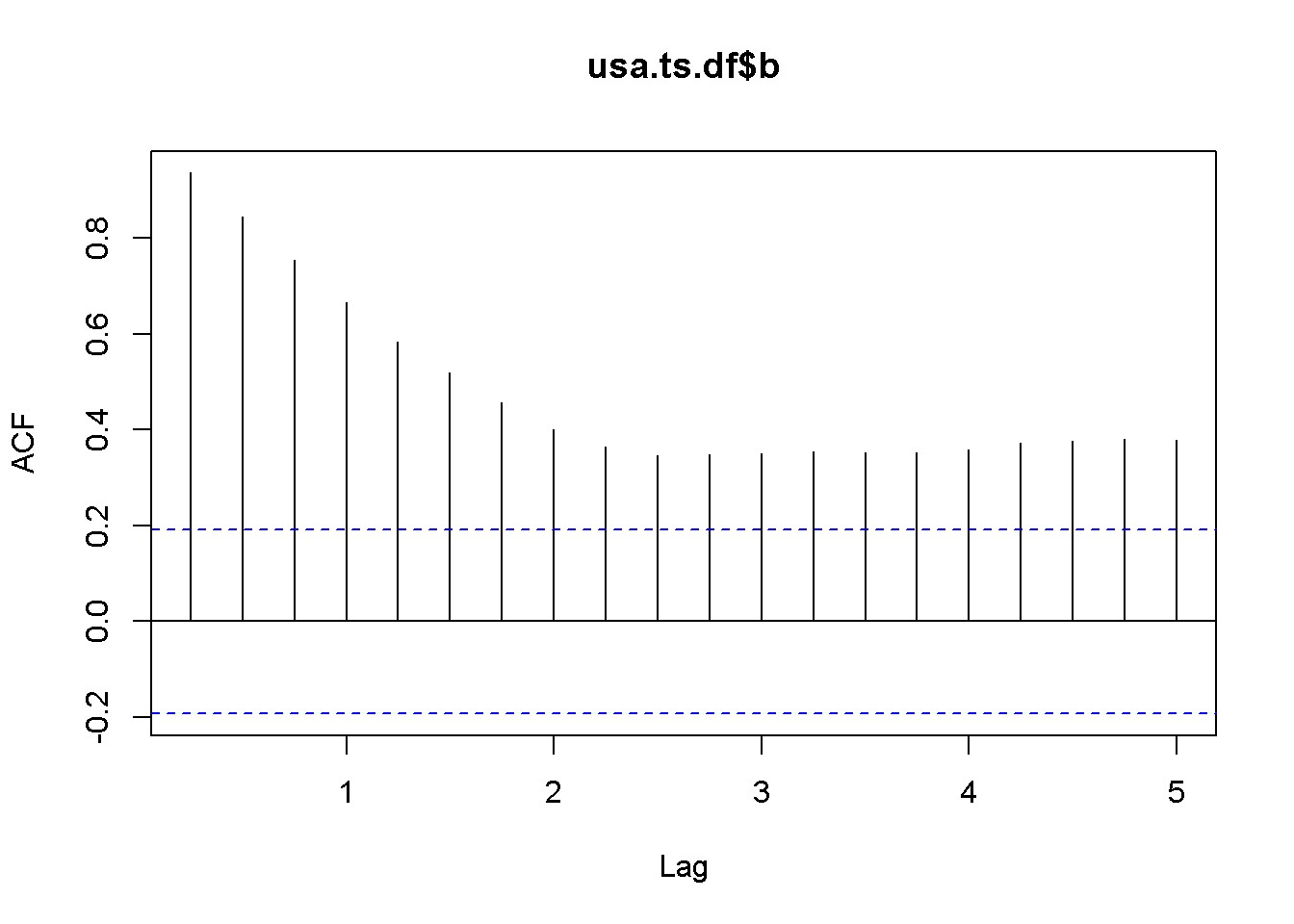
## alternative hypothesis: stationary

The result of the test is a *p*-value greater than our chosen significance level of 0.05; therefore, we cannot reject the null hypothesis of nonstationarity.

**plot**(usa.ts.df$b)



**Acf**(usa.ts.df$b)



**adf.test**(usa.ts.df$b, k=10)

##

## Augmented Dickey-Fuller Test

##

## data: usa.ts.df$b

## Dickey-Fuller = -2.984, Lag order = 10, p-value = 0.169 ## alternative hypothesis: stationary

Here is a code to reproduce the results in the textbook.

f <- usa.ts.df$f

f.dyn <- **dynlm**(**d**(f)~**L**(f)+**L**(**d**(f)))

**tidy**(f.dyn)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| term | | | estimate | | std.error | | statistic | | p.value  0.088337  0.013884  0.000000 | |
| (Intercept) | | | 0.172522 | | 0.100233 | | 1.72121 | |
| L(f) | | | -0.044621 | | 0.017814 | | -2.50482 | |
| L(d(f)) | | | 0.561058 | | 0.080983 | | 6.92812 | |
| b <- usa.ts.df$b  b.dyn <- **dynlm**(**d**(b)~**L**(b)+**L**(**d**(b)))  **tidy**(b.dyn) | | | | | | | | |  | |
| term | estimate | | std.error | | statistic | | p.value | |
| (Intercept) | 0.236873 | | 0.129173 | | 1.83376 | | 0.069693 | |
| L(b) | -0.056241 | | 0.020808 | | -2.70285 | | 0.008091 | |
| L(d(b)) | 0.290308 | | 0.089607 | | 3.23979 | | 0.001629 | |

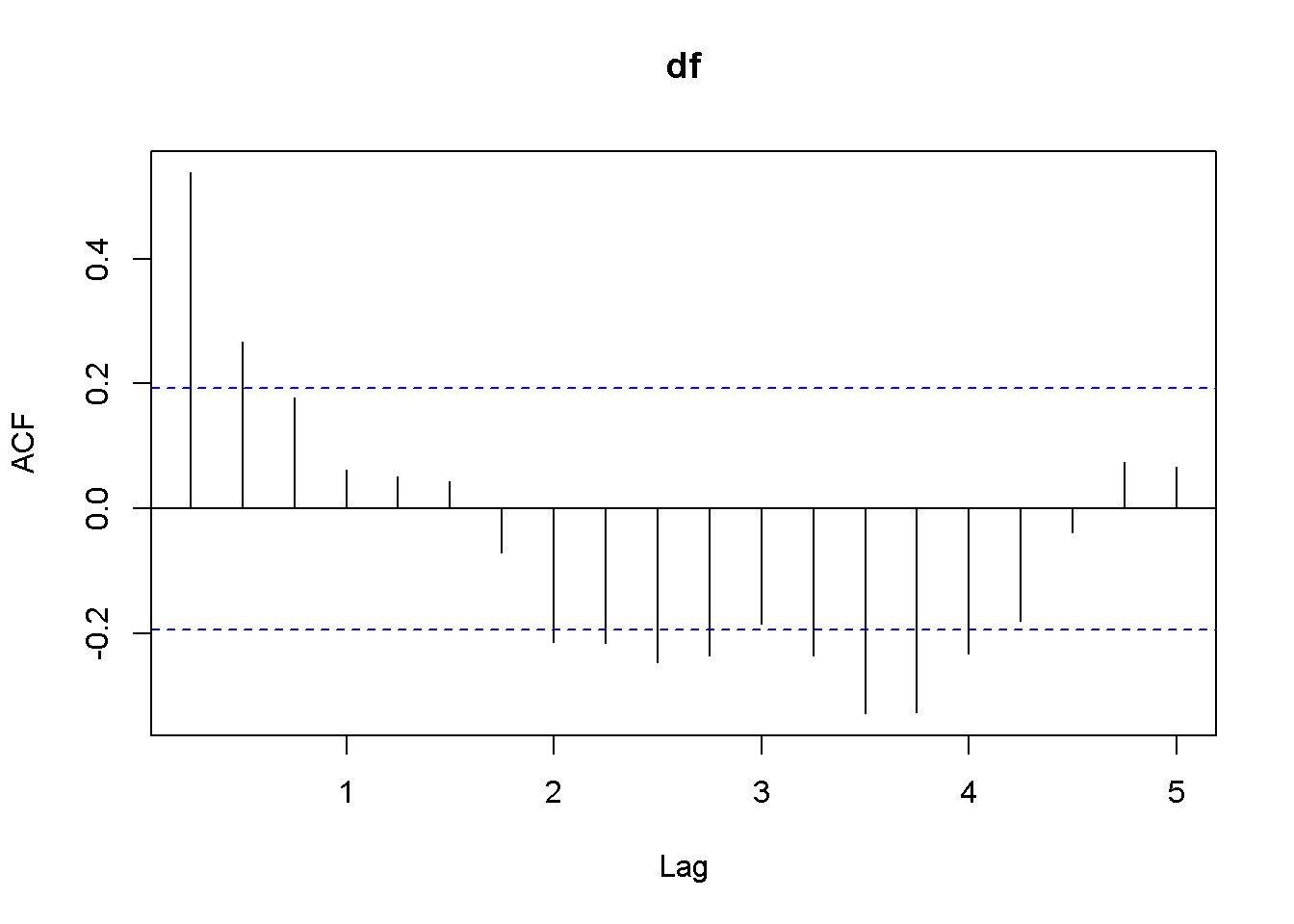
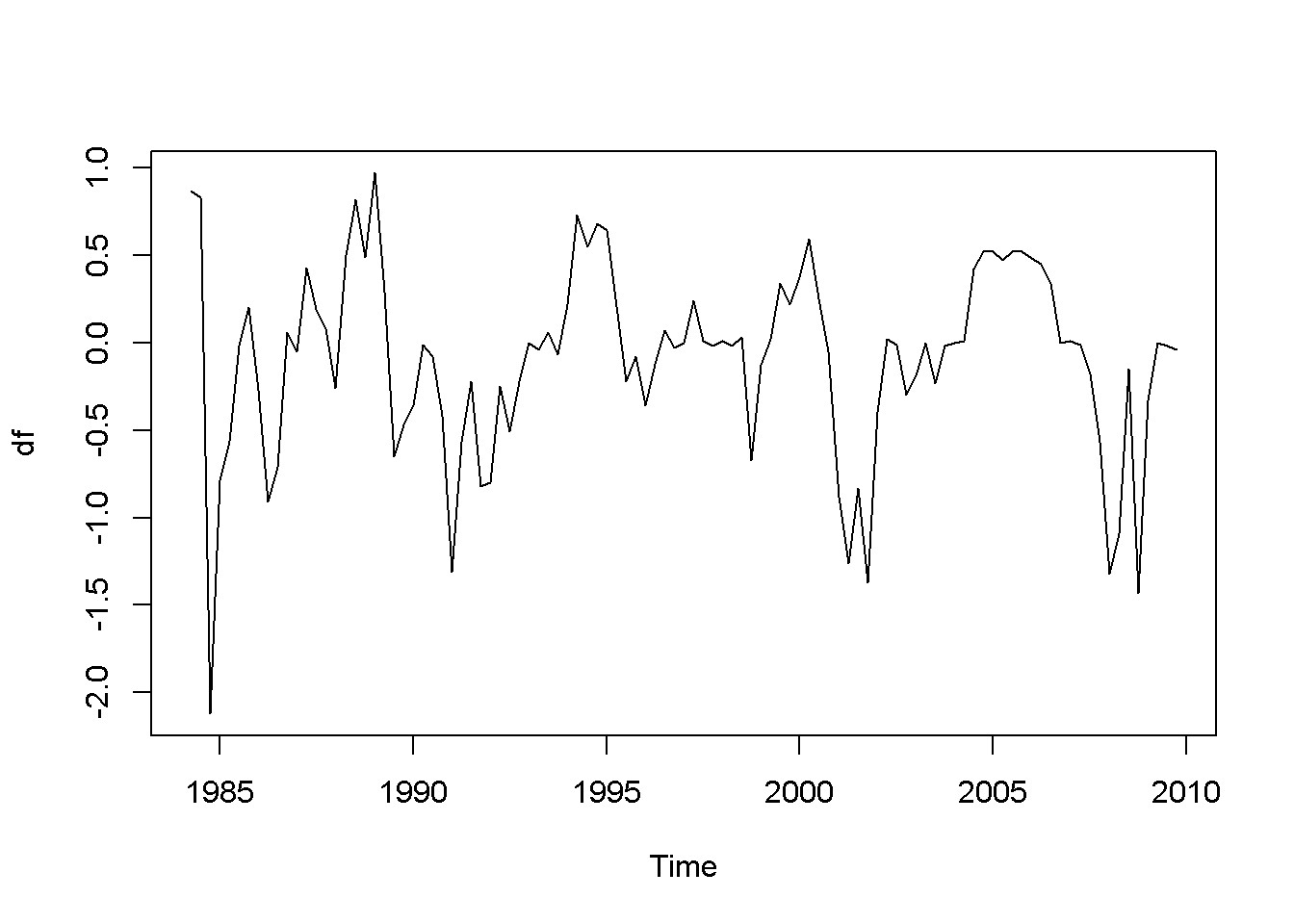


Figure 12.6: Plot and correlogram for series diff(f) in dataset usa

A concept that is closely related to stationarity is **order of integration**, which is how many times we need to difference a series untill it becomes stationary. A series is **I(0)**, that is, integrated of order 0 if it is already stationary (it is stationary *in levels*, not in differences); a series is **I(1)** if it is nonstationary in levels, but stationary in its first differences.

df <- **diff**(usa.ts.df$f)

**plot**(df) **Acf**(df) **adf.test**(df, k=2)

##

## Augmented Dickey-Fuller Test

##

## data: df

## Dickey-Fuller = -4.178, Lag order = 2, p-value = 0.01

## alternative hypothesis: stationary

db <- **diff**(usa.ts.df$b)

**plot**(db) **Acf**(db) **adf.test**(db, k=1)

##

## Augmented Dickey-Fuller Test

##

## data: db

## Dickey-Fuller = -6.713, Lag order = 1, p-value = 0.01

## alternative hypothesis: stationary

Both the plots and the DF tests indicate that the *f* and *b* series are stationary in first differences, which makes each of them integrated of order 1. The next code

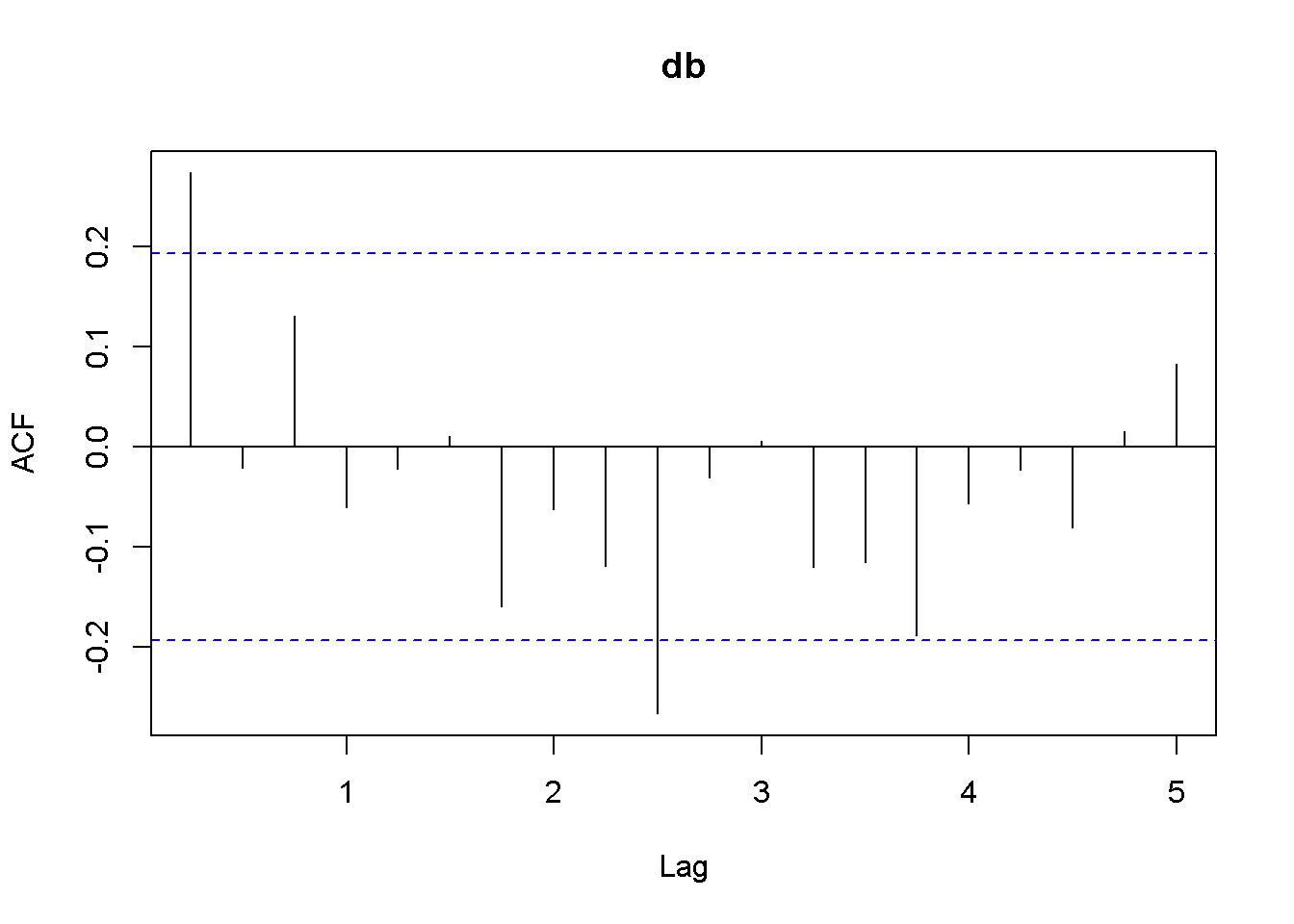
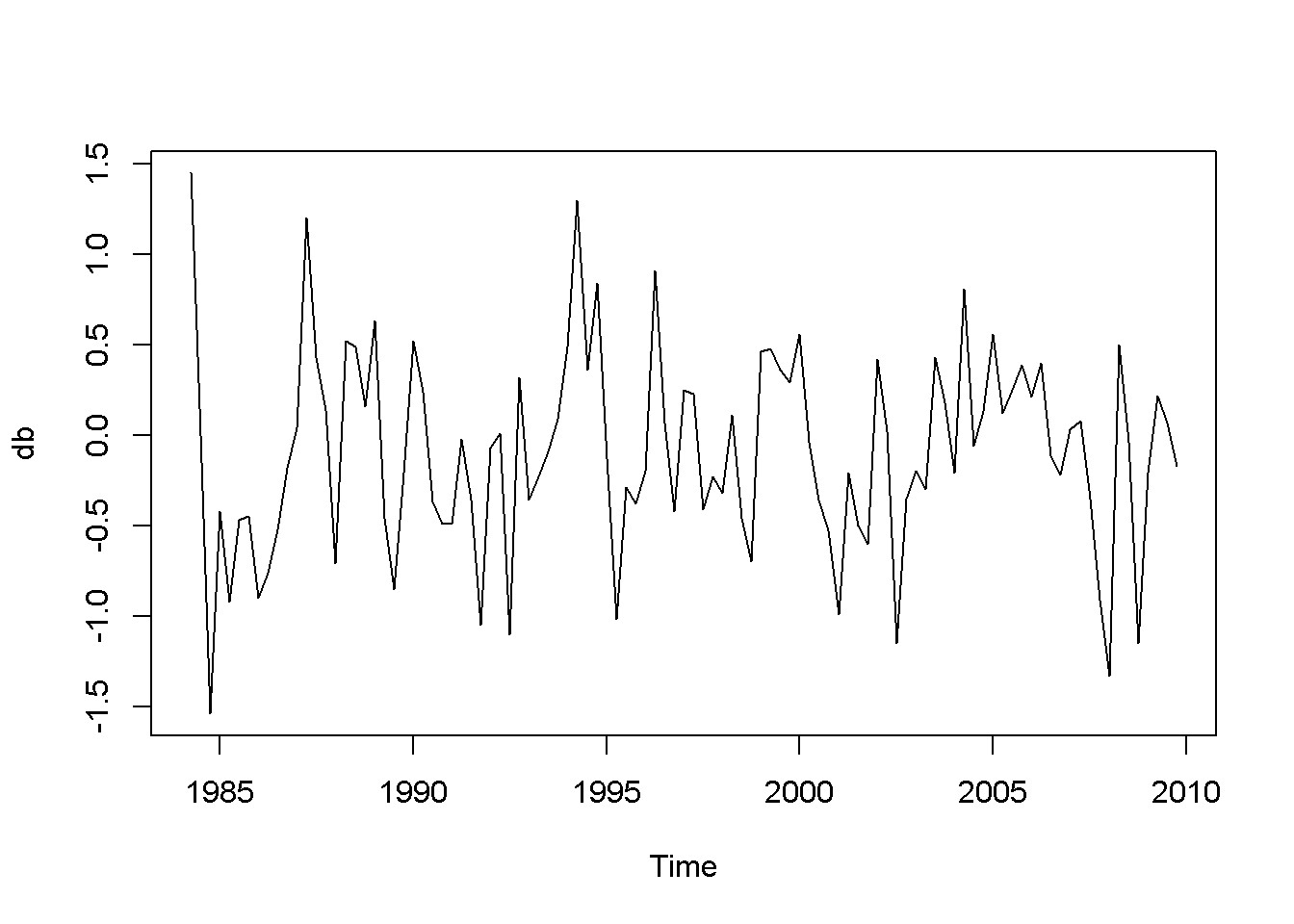


Figure 12.7: Plot and correlogram for series diff(b) in dataset usa

sequence reproduces the results in the textbook. Please note the term (*−*1) in the dynlm command; it tells *R* that we do not want an intercept in our model. Figures

12.6 and 12.7 show plots of the differenced *f* and *b* series, respectively.

df.dyn <- **dynlm**(**d**(df)~**L**(df)-1) db.dyn <- **dynlm**(**d**(db)~**L**(db)-1) **tidy**(df.dyn)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| term | | estimate | | std.error | | statistic | | p.value 0 | | |
| L(df) | | -0.446986 | | 0.081462 | | -5.48706 | |
| **tidy**(db.dyn) | | | |  | |  | |  | | |
| term | | estimate | | std.error | | statistic | | p.value |
| L(db) | | -0.701796 | | 0.091594 | | -7.66205 | | 0 |

Function ndiffs() in the package forecast is a very convenient way of determining the order of integration of a series. The arguments of this function are x, a time series, alpha, the significacnce level of the test (0.05 by default), test= one of “kpss”,

“adf”, or “pp”, which indicates the unit root test to be used; we have only studied the “adf” test.), and max.d= maximum number of differences. The output of this function is an integer, which is the order of integration of the time series.

**ndiffs**(f)

## [1] 1

**ndiffs**(b)

## [1] 1

As we have already found, the orders of integration for both *f* and *b* are 1.

Lab 4 Cointegration

Two series are cointegrated when their trends are not too far apart and are in some sense similar. This vague statement, though, can be made precise by conducting a cointegration test, which tests whether the residuals from regressing one series on the other one are stationary. If they are, the series are cointegrated. Thus, a cointegration test is in fact a Dickey-Fuler stationarity test on residuals, and its null hypothesis is of noncointegration. In other words, we would like to reject the null hypothesis in a cointegration test, as we wanted in a stationarity test.

Let us apply this method to determine the state of cointegration between the series *f* and *b* in dataset *usa*.

fb.dyn <- **dynlm**(b~f) ehat.fb <- **resid**(fb.dyn) **ndiffs**(ehat.fb) *#result: 1*

## [1] 1

output <- **dynlm**(**d**(ehat.fb)~**L**(ehat.fb)+**L**(**d**(ehat.fb))-1) *#no constant*

foo <- **tidy**(output) foo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | Statistic | p.value |
| L(ehat.fb) | -0.224509 | 0.053504 | -4.19613 | 0.000059 |
| L(d(ehat.fb)) | 0.254045 | 0.093701 | 2.71124 | 0.007891 |

The relevant statistic is *τ* = *−*4*.*196133, which is less than *−*3*.*37, the relevant critical value for the cointegration test. In conclusion, we reject the null hypothesis that the residuals have unit roots, therefore the series are cointegrated.

*R* has a special function to perform cointegration tests, function po.test in package tseries. (The name comes from the method it uses, which is called “Phillips-

Ouliaris.”) The main argument of the function is a matrix having in its first column the dependent variable of the cointegration equation and the independent variables in the other columns. Let me illustrate its application in the case of the same series *fb* and *f*.

bfx <- **as.matrix**(**cbind**(b,f), demean=FALSE) **po.test**(bfx)

##

## Phillips-Ouliaris Cointegration Test

##

## data: bfx

## Phillips-Ouliaris demeaned = -20.51, Truncation lag parameter = 1,

## p-value = 0.0499

The PO test marginally rejects the null of no cointegration at the 5 percent level.

## 12.5 The Error Correction Model

A relationship between cointegrated I(1) variables is a long run relationship, while a relationship between I(0) variables is a short run one. The short run error correction model combines, in some sense, short run and long run effects. Starting from an ARDL(1,1) model (Equation 12.6) and assuming that there is a steady state (long run) relationship between *y* and *x*, one can derive the **error correction** model in Equation 12.7, where more lagged differences of *x* may be necessary to eliminate autocorrelation.

*yt* = *δ* + *θ*1*yt−*1 + *δ*0*xt* + *δ*1*xt−*1 + *νt* (12.6)

∆*yt* = *−α*(*yt−*1 *− β*1 *− β*2*xt−*1) + *δ*0∆*xt* + *νt* (12.7)

In the case of the US bonds and funds example, the error correction model can be constructed as in Equation 12.8.

∆*bt* = *−α*(*bt−*1 *− β*1 *− β*2*ft−*1) + *δ*0∆*ft* + *δ*1∆*ft−*1 + *νt* (12.8)

The *R* function that estimates a nonlinear model such as the one in Equation 12.8 is nls, which requires three main argumants: a formula, which is the regression model to be estimated written using regular text mathematical operators, a start= list of guessed or otherwise approximated values of the estimated parameters to initiate a Gauss-Newton numerical optimization process, and data= a data frame, list, or environment data source. Please note that data cannot be a matrix.

In the next code sequence, the initial values of the parameters have been determined by estimating Equation 12.6 with *b* and *f* replacing *y* and *x*.

b.ols <- **dynlm**(**L**(b)~**L**(f)) b1ini <- **coef**(b.ols)[[1]] b2ini <- **coef**(b.ols)[[2]]

d.ols <- **dynlm**(b~**L**(b)+f+**L**(f)) aini <- 1-**coef**(d.ols)[[2]] d0ini <- **coef**(d.ols)[[3]] d1ini <- **coef**(d.ols)[[4]]

Db <- **diff**(b)

Table 12.2: Parameter estimates in the error correction model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| a | 0.141877 | 0.049656 | 2.85720 | 0.005230 |
| b1 | 1.429188 | 0.624625 | 2.28807 | 0.024304 |
| b2 | 0.776557 | 0.122475 | 6.34052 | 0.000000 |
| d0 | 0.842463 | 0.089748 | 9.38697 | 0.000000 |
| d1 | -0.326845 | 0.084793 | -3.85463 | 0.000208 |

Table 12.3: Stationarity test within the error correction model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| L(ehat) | -0.168488 | 0.042909 | -3.92668 | 0.000158 |
| L(d(ehat)) | 0.179486 | 0.092447 | 1.94149 | 0.055013 |
| Df <- **diff**(f)  Lb <- **lag**(b,-1)  Lf <- **lag**(f,-1) LDf <- **lag**(**diff**(f),-1)  bfset <- **data.frame**(**ts.union**(**cbind**(b,f,Lb,Lf,Db,Df,LDf))) formula <- Db ~ -a\*(Lb-b1-b2\*Lf)+d0\*Df+d1\*LDf bf.nls <- **nls**(formula, na.action=na.omit, data=bfset, start=**list**(a=aini, b1=b1ini, b2=b2ini, d0=d0ini, d1=d1ini))  **kable**(**tidy**(bf.nls),  caption="Parameter estimates in the error correction model") | | | | | | |

The error correction model can also be used to test the two series for cointegration. All we need to do is to test the errors of the correction part embedded in Equation 12.8 for stationarity. The estimated errors are given by Equation 12.9.

*e*ˆ*t−*1 = *bt−*1 *− β*1 *− β*2*ft−*1 (12.9)

ehat <- bfset$Lb-**coef**(bf.nls)[[2]]-**coef**(bf.nls)[[3]]\*bfset$Lf ehat <- **ts**(ehat)

ehat.adf <- **dynlm**(**d**(ehat)~**L**(ehat)+**L**(**d**(ehat))-1)

**kable**(**tidy**(ehat.adf),

caption="Stationarity test within the error correction model")

foo <- **tidy**(ehat.adf)

To test for cointegration, one should compare the *t*-ratio of the lagged term shown as ‘statistic’ in Equation 12.3, *t* = *−*3*.*927 to the critical value of *−*3*.*37. The result is to reject the null of no cointegration, which means the series are cointegrated.

Lab 5 - VEC and VAR Models

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!* **library**(tseries) *# for `adf.test()`* **library**(dynlm) *#for function `dynlm()`* **library**(vars) *# for function `VAR()`*  **library**(nlWaldTest) *# for the `nlWaldtest()` function* **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 datasets* **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for `kable()`* **library**(forecast) |

New package: vars (Pfaff 2013).

When there is no good reason to assume a one-way causal relationship between two time series variables we may think of their relationship as one of mutual interaction. The concept of “vector,” as in vector error correction refers to a number of series in such a model.

## 13.1 VAR and VEC Models

Equations 13.1 and 13.1 show a generic **vector autoregression** model of order 1, VAR(1), which can be estimated if the series are both I(0). If they are I(1), the same equations need to be estimated in first differences.

195

*y*

|  |  |
| --- | --- |
| *yt* = *β*10 + *β*11*yt−*1 + *β*12*xt−*1 + *νt* | (13.1) |
| *xt* = *β*20 + *β*21*yt−*1 + *β*22*xt−*1 + *νtx* | (13.2) |

If the two variables in Equations 13.1 and 13.2 and are cointegrated, their cointegration relationship should be taken into account in the model, since it is valuable information; such a model is called **vector error correction**. The cointegration relationship is, remember, as shown in Equation 13.3, where the error term has been proven to be stationary.

*yt* = *β*0 + *β*1*xt* + *et* (13.3)

## 13.2 Estimating a VEC Model

The simplest method is a two-step procedure. First, estimate the cointegrating relationship given in Equation 13.3 and created the lagged resulting residual series *e*ˆ*t−*1 = *yt−*1 *− b*0 *− b*1*xt−*1. Second, estimate Equations 13.4 and 13.5 by OLS.

|  |  |
| --- | --- |
| ∆*yt* = *α*10 + *α*11 + *e*ˆ*t−*1 + *νty* | (13.4) |
| ∆*xt* = *α*20 + *α*21 + *e*ˆ*t−*1 + *νtx* | (13.5) |

The following example uses the dataset *gdp*, which includes GDP series for Australia and USA for the period since 1970:1 to 2000:4. First we determine the order of integration of the two series.

|  |
| --- |
| **data**("gdp", package="PoEdata") gdp <- **ts**(gdp, start=**c**(1970,1), end=**c**(2000,4), frequency=4)  **ts.plot**(gdp[,"usa"],gdp[,"aus"], type="l", lty=**c**(1,2), col=**c**(1,2))  **legend**("topleft", border=NULL, legend=**c**("USA","AUS"),  lty=**c**(1,2), col=**c**(1,2)) |

Figure 13.1 represents the two series in levels, revealing a common trend and, therefore, suggesting that the series are nonstationary.

*13.2. ESTIMATING A VEC MODEL*

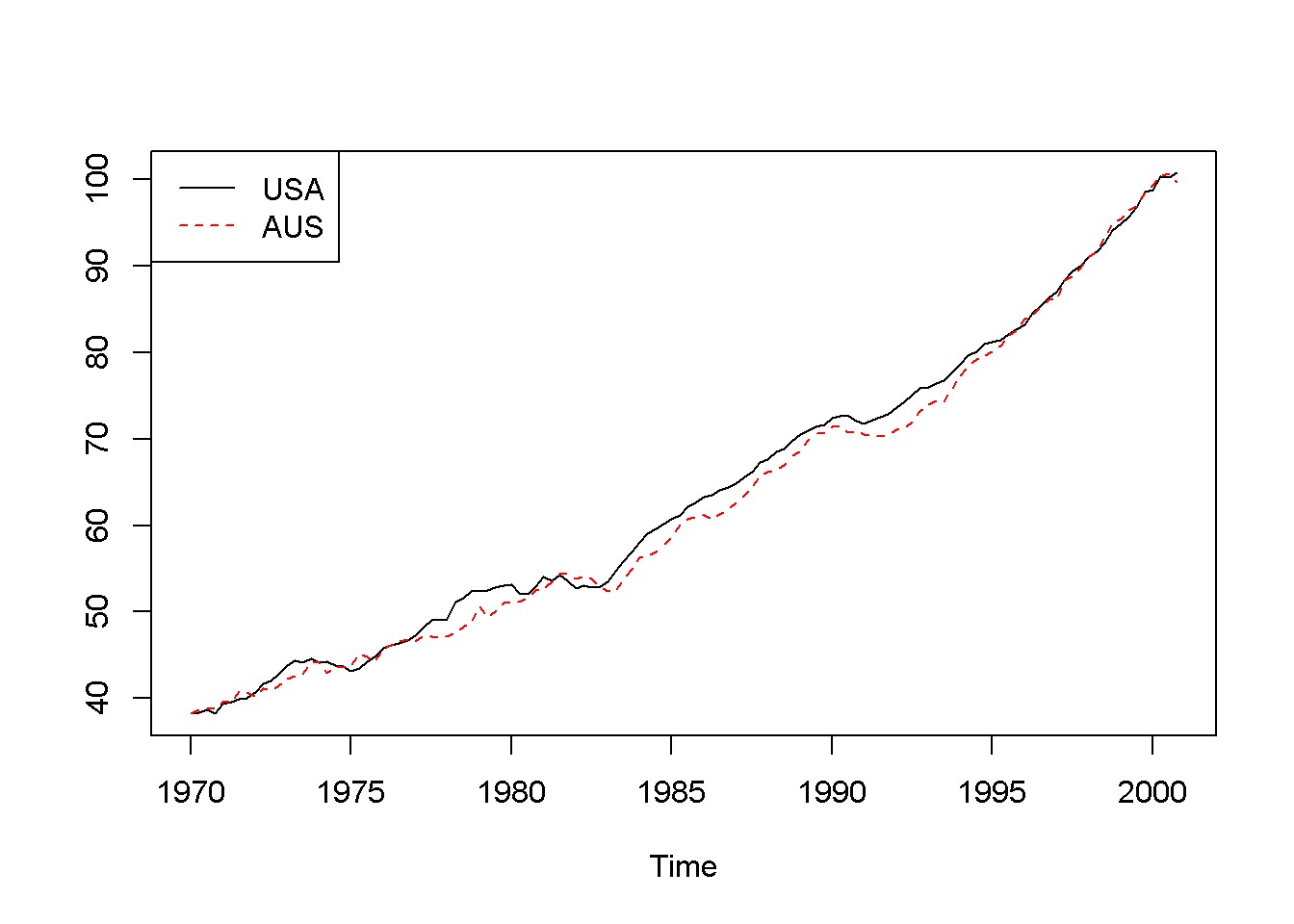


Figure 13.1: Australian and USA GDP series from dataset ’gdp’

**adf.test**(gdp[,"usa"])

##

## Augmented Dickey-Fuller Test

##

## data: gdp[, "usa"]

## Dickey-Fuller = -0.9083, Lag order = 4, p-value = 0.949

## alternative hypothesis: stationary

**adf.test**(gdp[,"aus"])

##

## Augmented Dickey-Fuller Test

##

## data: gdp[, "aus"]

## Dickey-Fuller = -0.6124, Lag order = 4, p-value = 0.975

## alternative hypothesis: stationary

**adf.test**(**diff**(gdp[,"usa"]))

##

## Augmented Dickey-Fuller Test

##

Table 13.1: The results of the cointegration equation ’cint1.dyn’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| usa | 0.985 | 0.002 | 594.787 | 0 |

## data: diff(gdp[, "usa"])

## Dickey-Fuller = -4.293, Lag order = 4, p-value = 0.01

## alternative hypothesis: stationary

**adf.test**(**diff**(gdp[,"aus"]))

##

## Augmented Dickey-Fuller Test

##

## data: diff(gdp[, "aus"])

## Dickey-Fuller = -4.417, Lag order = 4, p-value = 0.01

## alternative hypothesis: stationary

The stationarity tests indicate that both series are I(1), Let us now test them for cointegration, using Equations 13.6 and 13.7.

|  |  |
| --- | --- |
| *aust* = *β*1*usat* + *et* | (13.6) |
| *e*ˆ*t* = *aust − β*1*usat* | (13.7) |

cint1.dyn <- **dynlm**(aus~usa-1, data=gdp) **kable**(**tidy**(cint1.dyn), digits=3,

caption="The results of the cointegration equation 'cint1.dyn'")

ehat <- **resid**(cint1.dyn) cint2.dyn <- **dynlm**(**d**(ehat)~**L**(ehat)-1) **summary**(cint2.dyn)

##

## Time series regression with "ts" data:

## Start = 1970(2), End = 2000(4)

##

## Call:

## dynlm(formula = d(ehat) ~ L(ehat) - 1)

##

## Residuals:

Min 1Q Median 3Q Max

## -1.4849 -0.3370 -0.0038 0.4656 1.3507

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## L(ehat) -0.1279 0.0443 -2.89 0.0046 \*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 0.598 on 122 degrees of freedom

## Multiple R-squared: 0.064, Adjusted R-squared: 0.0564 ## F-statistic: 8.35 on 1 and 122 DF, p-value: 0.00457

Our test rejects the null of no cointegration, meaning that the series are cointegrated. With cointegrated series we can construct a VEC model to better understand the causal relationship between the two variables.

vecaus<- **dynlm**(**d**(aus)~**L**(ehat), data=gdp) vecusa <- **dynlm**(**d**(usa)~**L**(ehat), data=gdp) **tidy**(vecaus)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| term | | | estimate | | std.error | | statistic | p.value  0.000000  0.039893 | | |
| (Intercept) | | | 0.491706 | | 0.057909 | | 8.49094 |
| L(ehat) | | | -0.098703 | | 0.047516 | | -2.07727 |
| **tidy**(vecusa) | | | | |  | |  |  | | |
| term | estimate | | std.error | | statistic | | | p.value |
| (Intercept) | 0.509884 | | 0.046677 | | 10.923715 | | | 0.000000 |
| L(ehat) | 0.030250 | | 0.038299 | | 0.789837 | | | 0.431168 |

The coefficient on the error correction term (*e*ˆ*t−*1) is significant for Australia, suggesting that changes in the US economy do affect Australian economy; the error correction coefficient in the US equation is not statistically significant, suggesting that changes in Australia do not influence American economy. To interpret the sign of the error correction coefficient, one should remember that *e*ˆ*t−*1 measures the deviation of Australian economy from its cointegrating level of 0*.*985 of the US economy (see Equations 13.6 and 13.7 and the value of *β*1 in Table 13.1).

Lab 6 - Estimating a VAR Model

The VAR model can be used when the variables under study are I(1) but not cointegrated. The model is the one in Equations **??**, but in differences, as specified in Equations 13.8 and 13.9.

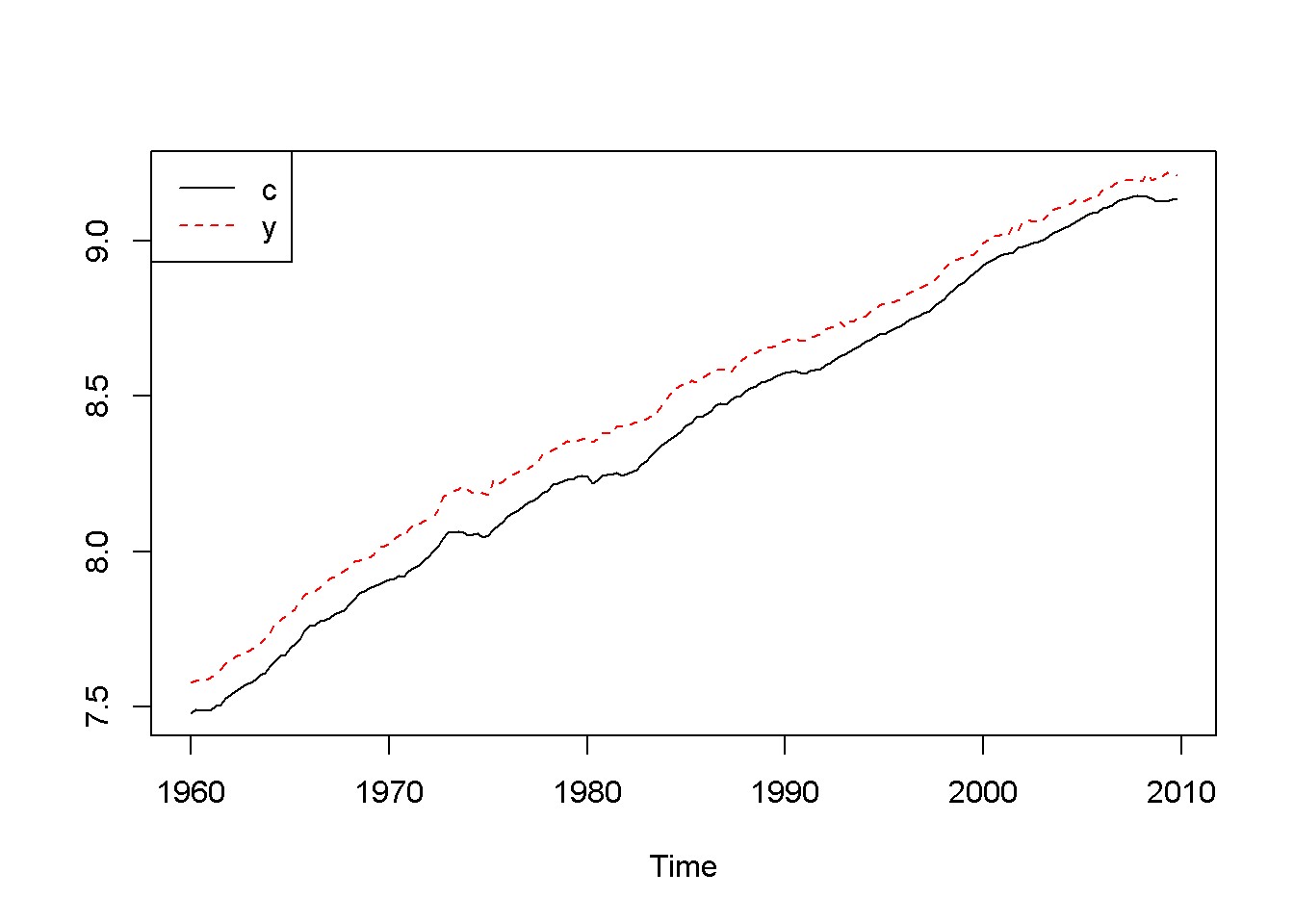


Figure 13.2: Logs of income (y) and consumption (c), dataset ’fred’

|  |  |
| --- | --- |
| ∆*yt* = *β*11∆*yt−*1 + *β*12∆*xt−*1 + *νt*∆*y* | (13.8) |
| ∆*xt* = *β*21∆*yt−*1 + *β*22∆*xt−*1 + *νt*∆*x* | (13.9) |

Let us look at the income-consumption relationship based on the *fred* detaset, where consumption and income are already in logs, and the period is 1960:1 to 2009:4. Figure 13.2 shows that the two series both have a trend.

|  |
| --- |
| **data**("fred", package="PoEdata")  fred <- **ts**(fred, start=**c**(1960,1),end=**c**(2009,4),frequency=4) **ts.plot**(fred[,"c"],fred[,"y"], type="l", lty=**c**(1,2), col=**c**(1,2))  **legend**("topleft", border=NULL, legend=**c**("c","y"),  lty=**c**(1,2), col=**c**(1,2)) |

Are the two series cointegrated?

**Acf**(fred[,"c"]) **Acf**(fred[,"y"]) **adf.test**(fred[,"c"]) ## Augmented Dickey-Fuller Test

##

## data: fred[, "c"]

## Dickey-Fuller = -2.62, Lag order = 5, p-value = 0.316

## alternative hypothesis: stationary

**adf.test**(fred[,"y"])

##

## Augmented Dickey-Fuller Test

##

## data: fred[, "y"]

## Dickey-Fuller = -2.291, Lag order = 5, p-value = 0.454

## alternative hypothesis: stationary

**adf.test**(**diff**(fred[,"c"]))

##

## Augmented Dickey-Fuller Test

##

## data: diff(fred[, "c"])

## Dickey-Fuller = -4.713, Lag order = 5, p-value = 0.01

## alternative hypothesis: stationary

**adf.test**(**diff**(fred[,"y"]))

##

## Augmented Dickey-Fuller Test

##

## data: diff(fred[, "y"])

## Dickey-Fuller = -5.775, Lag order = 5, p-value = 0.01

## alternative hypothesis: stationary

cointcy <- **dynlm**(c~y, data=fred) ehat <- **resid**(cointcy) **adf.test**(ehat)

##

## Augmented Dickey-Fuller Test

##

## data: ehat

## Dickey-Fuller = -2.562, Lag order = 5, p-value = 0.341

## alternative hypothesis: stationary

Figure 13.3 shows a long serial correlation sequence; therefore, I will let *R* calculate the lag order in the ADF test. As the results of the above adf and cointegration

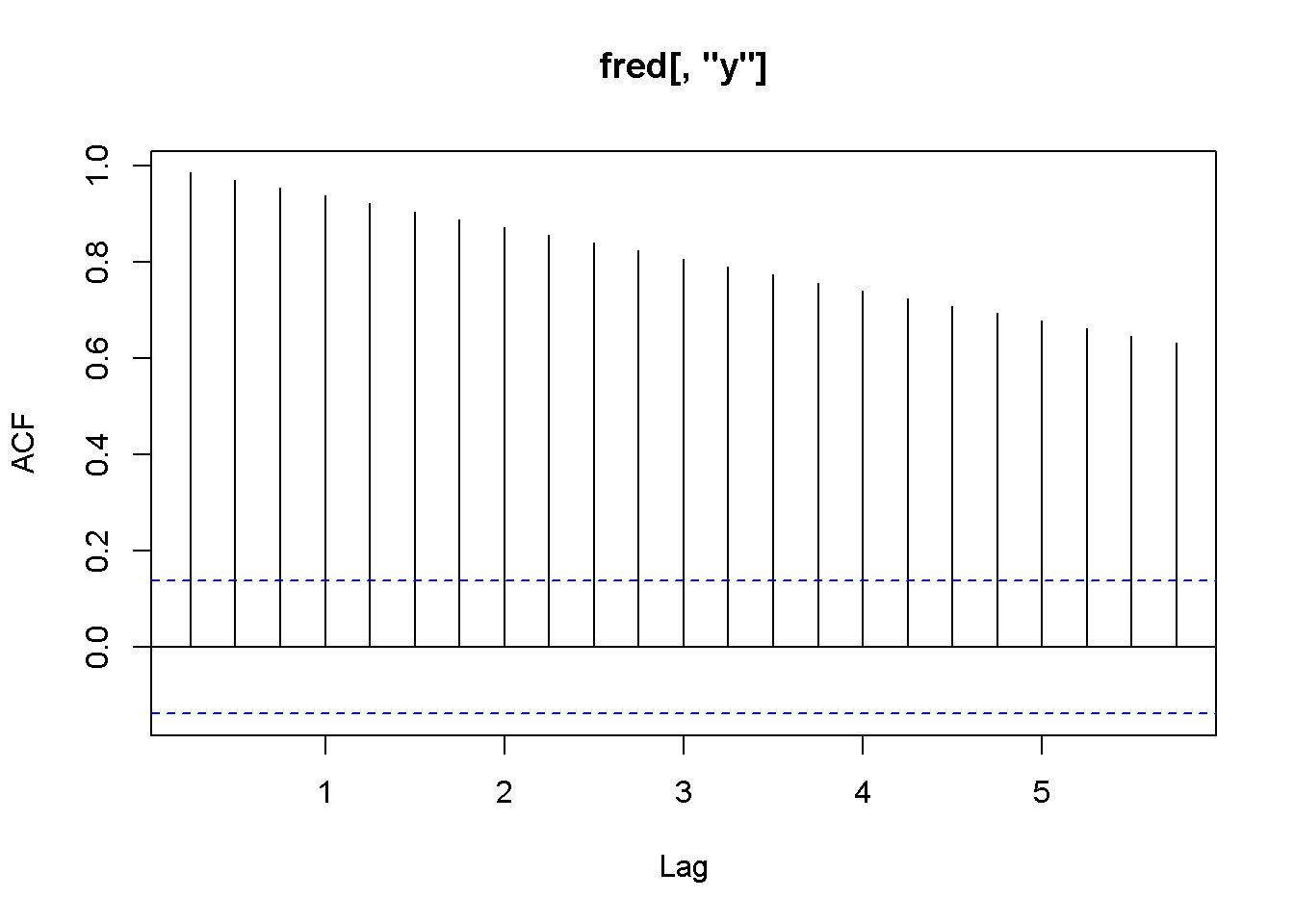
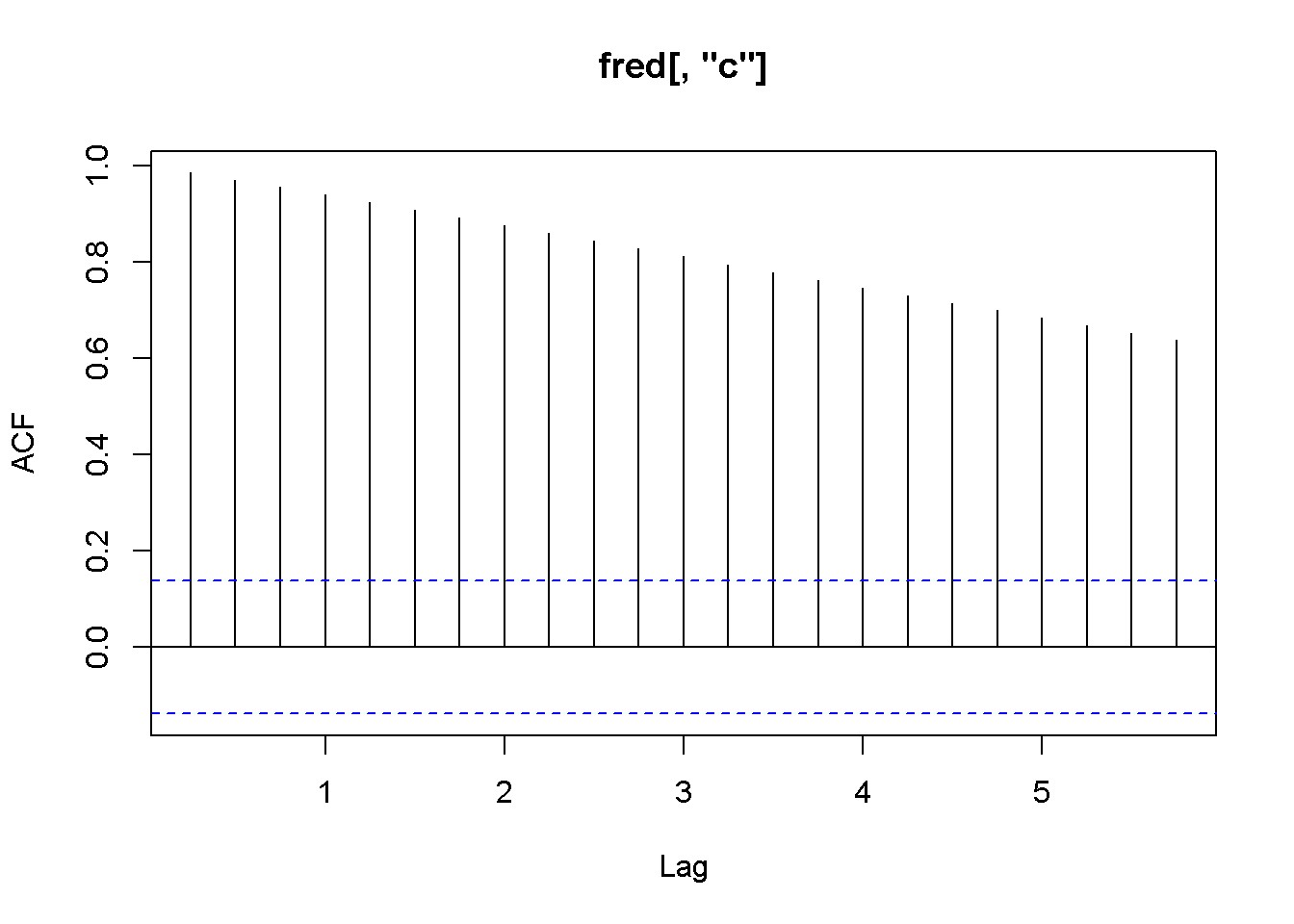


Figure 13.3: Correlograms for the series c and y, dataset fred

tests show, the series are both I(1) but they fail the cointegration test (the series are not cointegrated.) (Plese rememebr that the adf.test function uses a constant and trend in the test equation; therefore, the critical values are not the same as in the textbook. However, the results of the tests should be the same most of the time.)

**library**(vars)

Dc <- **diff**(fred[,"c"]) Dy <- **diff**(fred[,"y"]) varmat <- **as.matrix**(**cbind**(Dc,Dy))

varfit <- **VAR**(varmat) *# `VAR()` from package `vars`* **summary**(varfit)

##

## VAR Estimation Results:

## =========================

## Endogenous variables: Dc, Dy

## Deterministic variables: const

## Sample size: 198

## Log Likelihood: 1400.444

## Roots of the characteristic polynomial: ## 0.344 0.343 ## Call:

## VAR(y = varmat)

##

##

## Estimation results for equation Dc: ## ===================================

## Dc = Dc.l1 + Dy.l1 + const

##

Estimate Std. Error t value Pr(>|t|)

## Dc.l1 0.215607 0.074749 2.88 0.0044 \*\*

## Dy.l1 0.149380 0.057734 2.59 0.0104 \*

## const 0.005278 0.000757 6.97 4.8e-11 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

##

## Residual standard error: 0.00658 on 195 degrees of freedom

## Multiple R-Squared: 0.12, Adjusted R-squared: 0.111

## F-statistic: 13.4 on 2 and 195 DF, p-value: 3.66e-06

##

##

## Estimation results for equation Dy: ## ===================================

## Dy = Dc.l1 + Dy.l1 + const

##

## Estimate Std. Error t value Pr(>|t|)

## Dc.l1 0.475428 0.097326 4.88 2.2e-06 \*\*\*

## Dy.l1 -0.217168 0.075173 -2.89 0.0043 \*\*

## const 0.006037 0.000986 6.12 5.0e-09 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

##

## Residual standard error: 0.00856 on 195 degrees of freedom

## Multiple R-Squared: 0.112, Adjusted R-squared: 0.103

## F-statistic: 12.3 on 2 and 195 DF, p-value: 9.53e-06

##

##

##

## Covariance matrix of residuals:

## Dc Dy ## Dc 0.0000432 0.0000251

## Dy 0.0000251 0.0000733

##

## Correlation matrix of residuals:

## Dc Dy ## Dc 1.000 0.446

## Dy 0.446 1.000

Function VAR(), which is part of the package vars (Pfaff 2013), accepts the following main arguments: y= a matrix containing the endogenous variables in the VAR model, p= the desired lag order (default is 1), and exogen= a matrix of exogenous variables.

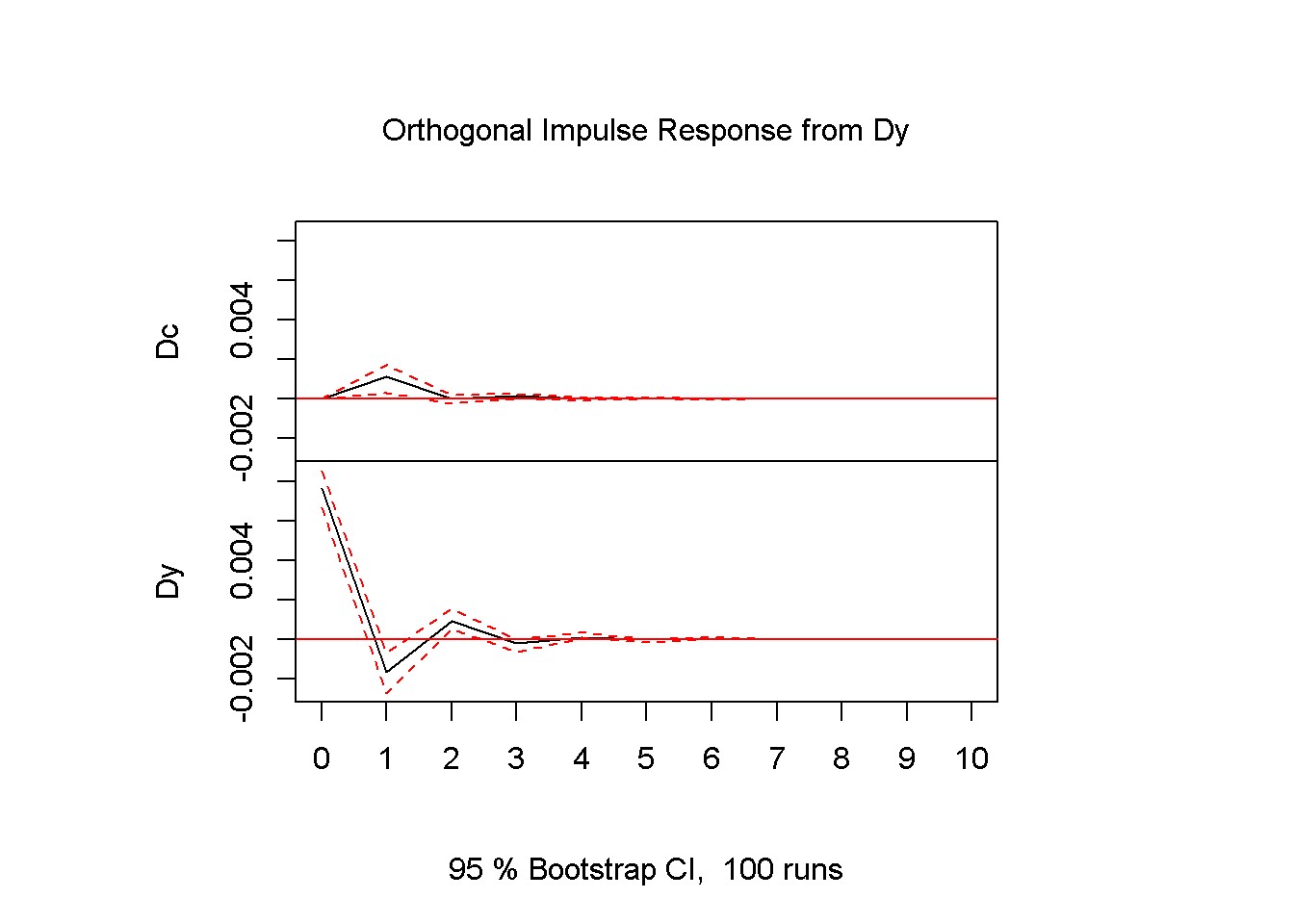
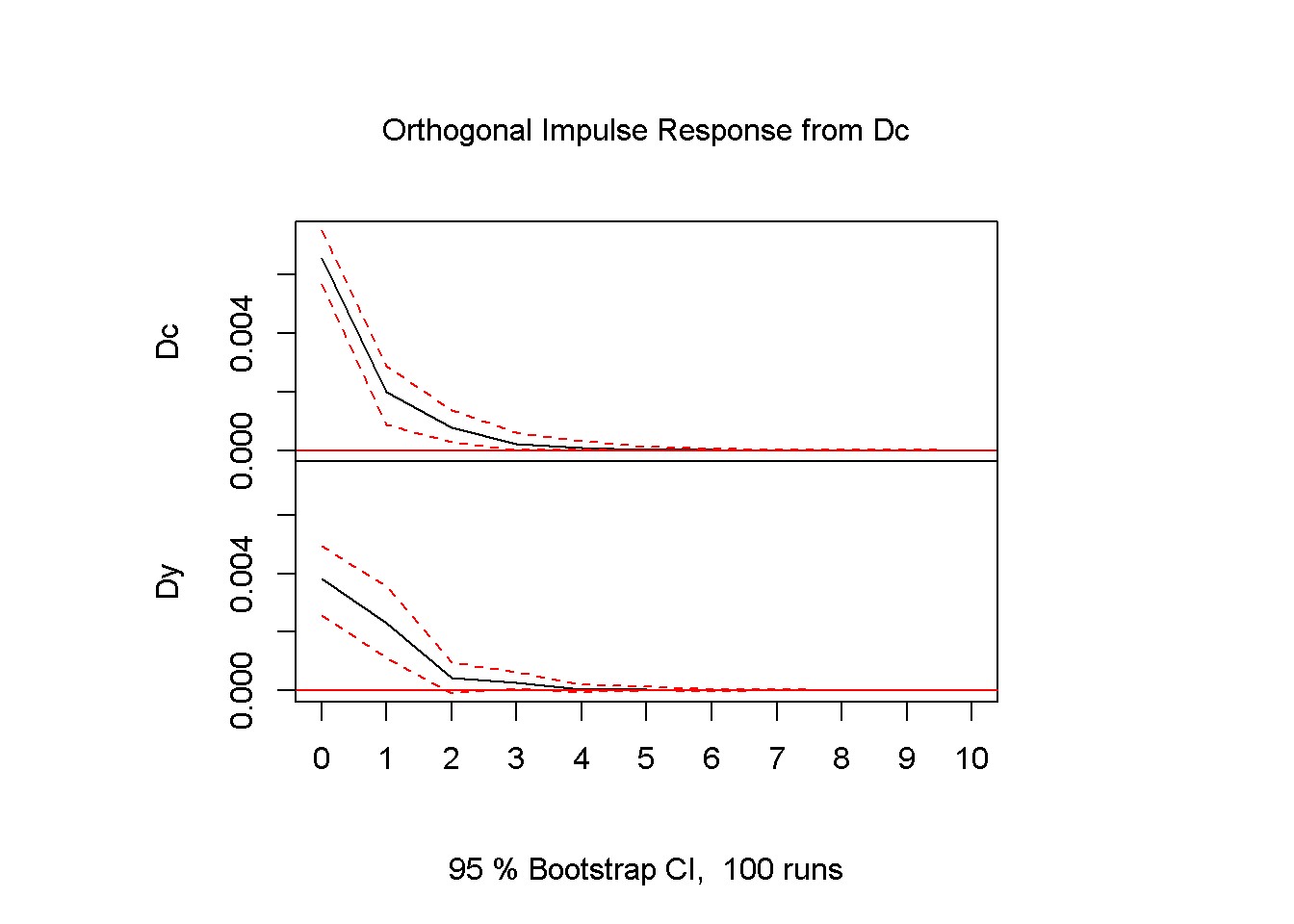


Figure 13.4: Impulse response diagrams for the series c and y, dataset fred

(VAR is a more powerful instrument than I imply here; please type ?VAR() for more information.) The results of a VAR model are more useful in analysing the time response to shocks in the variables, which is the topic of the next section.

## 13.4 Impulse Responses and Variance Decompositions

Impulse responses are best represented in graphs showing the responses of a VAR endogenous variable in time.

impresp <- **irf**(varfit) **plot**(impresp)

The interpretation of Figures 13.4 is straightforward: an impulse (shock) to *Dc* at time zero has large effects the next period, but the effects become smaller and smaller as the time passes. The dotted lines show the 95 percent interval estimates of these effects. The VAR function prints the values corresponding to the impulse response graphs.

**plot**(**fevd**(varfit)) *# `fevd()` is in package `vars`*

Forecast variance decomposition estimates the contribution of a shock in each variable to the response in both variables. Figure 13.5 shows that almost 100 percent of the variance in *Dc* is caused by *Dc* itself, while only about 80 percent in the variance of *Dy* is caused by *Dy* and the rest is caused by *Dc*. The *R* function fevd() in package vars allows forecast variance decomposition.

*13.4. IMPULSE RESPONSES AND VARIANCE DECOMPOSITIONS*

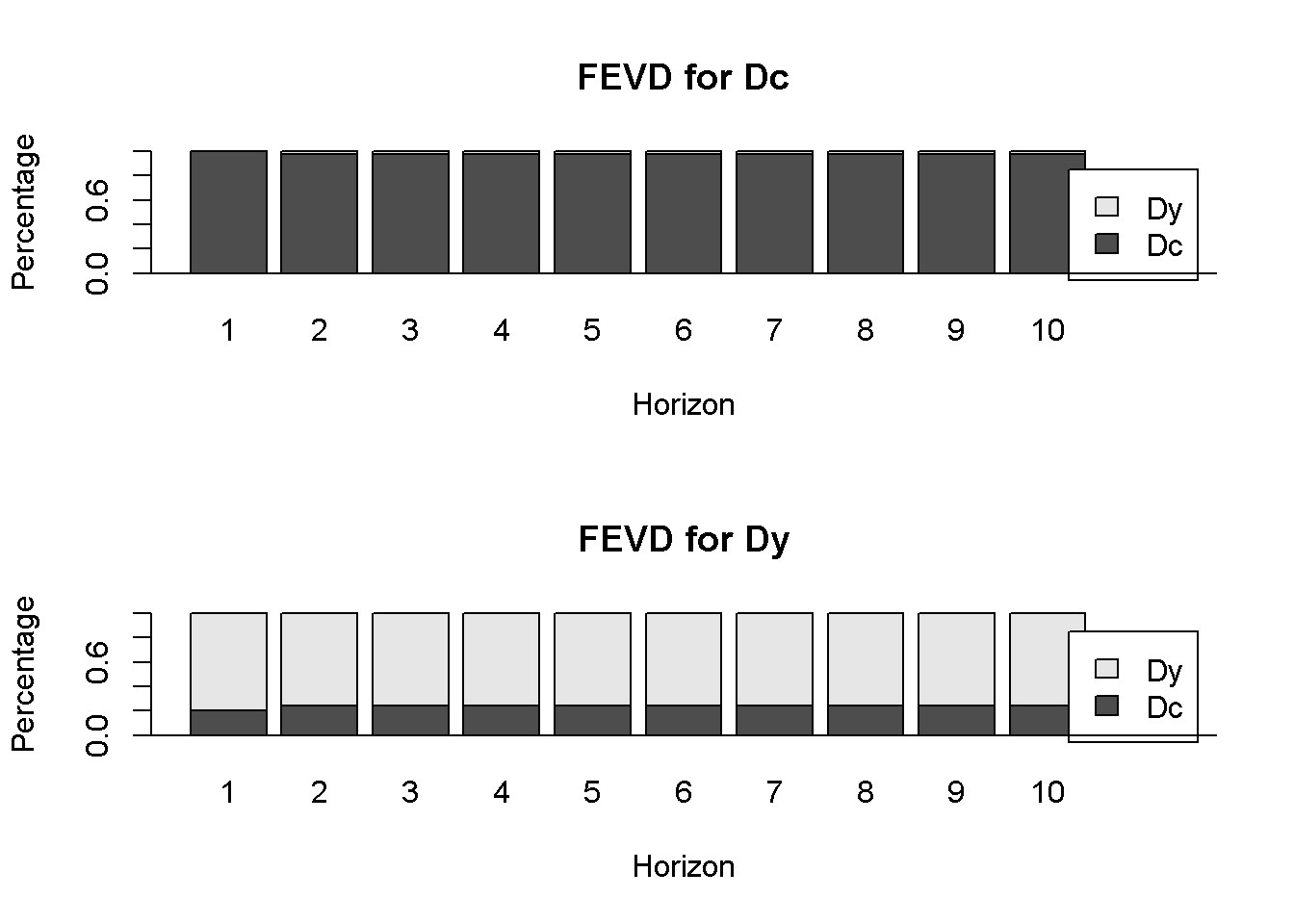


Figure 13.5: Forecast variance decomposition for the series c and y, dataset fred 206 *CHAPTER 13. VEC AND VAR MODELS*

Optional Lab 1 - Time-Varying Volatility and ARCH Models

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!* **library**(FinTS) *#for function `ArchTest()`* **library**(rugarch) *#for GARCH models* **library**(tseries) *# for `adf.test()`* **library**(dynlm) *#for function `dynlm()`* **library**(vars) *# for function `VAR()`*  **library**(nlWaldTest) *# for the `nlWaldtest()` function* **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 datasets* **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for `kable()`* **library**(forecast) |

New packages: FinTS (Graves 2014) and rugarch (Ghalanos 2015).

The **autoregressive conditional heteroskedasticity** (ARCH) model concerns time series with time-varying heteroskedasticity, where variance is conditional on the information existing at a given point in time.

207

## 14.1 The ARCH Model

The ARCH model assumes that the conditional mean of the error term in a time series model is constant (zero), unlike the nonstationary series we have discussed so far), but its conditional variance is not. Such a model can be described as in Equations 14.1, 14.2 and 14.3.

|  |  |
| --- | --- |
| *yt* = *φ* + *et* | (14.1) |
| *et|It−*1 *∼ N*(0*,ht*) | (14.2) |

*ht*  (14.3)

Equations 14.4 and 14.5 give both the test model and the hypotheses to **test for**

**ARCH effects** in a time series, where the residuals *e*ˆ*t* come from regressing the variable *yt* on a constant, such as 14.1, or on a constant plus other regressors; the test shown in Equation 14.4 may include several lag terms, in which case the null hypothesis (Equation 14.5) would be that all of them are jointly insignificant.

|  |  |
| --- | --- |
| *e*ˆ2*t* = *γ*0 + *γ*1*e*ˆ2*t−*1 + *...* + *γqe*2*t−q* + *νt* | (14.4) |
| *H*0 : *γ*1 = *...* = *γq* = 0 *HA* : *γ*1 = 0*6* *or ...γq* = 0*6* | (14.5) |

The null hypothesis is that there are no ARCH effects. The test statistic is



. The following example uses the dataset *byd*, which contains 500 generated observations on the returns to shares in BrightenYourDay Lighting. Figure 14.1 shows a time series plot of the data and histogram.

|  |
| --- |
| **data**("byd", package="PoEdata")  rTS <- **ts**(byd$r) **plot.ts**(rTS)  **hist**(rTS, main="", breaks=20, freq=FALSE, col="grey") |

Let us first perform, step by step, the ARCH test described in Equations 14.4 and

14.5, on the variable *r* from dataset *byd*.

*14.1. THE ARCH MODEL*

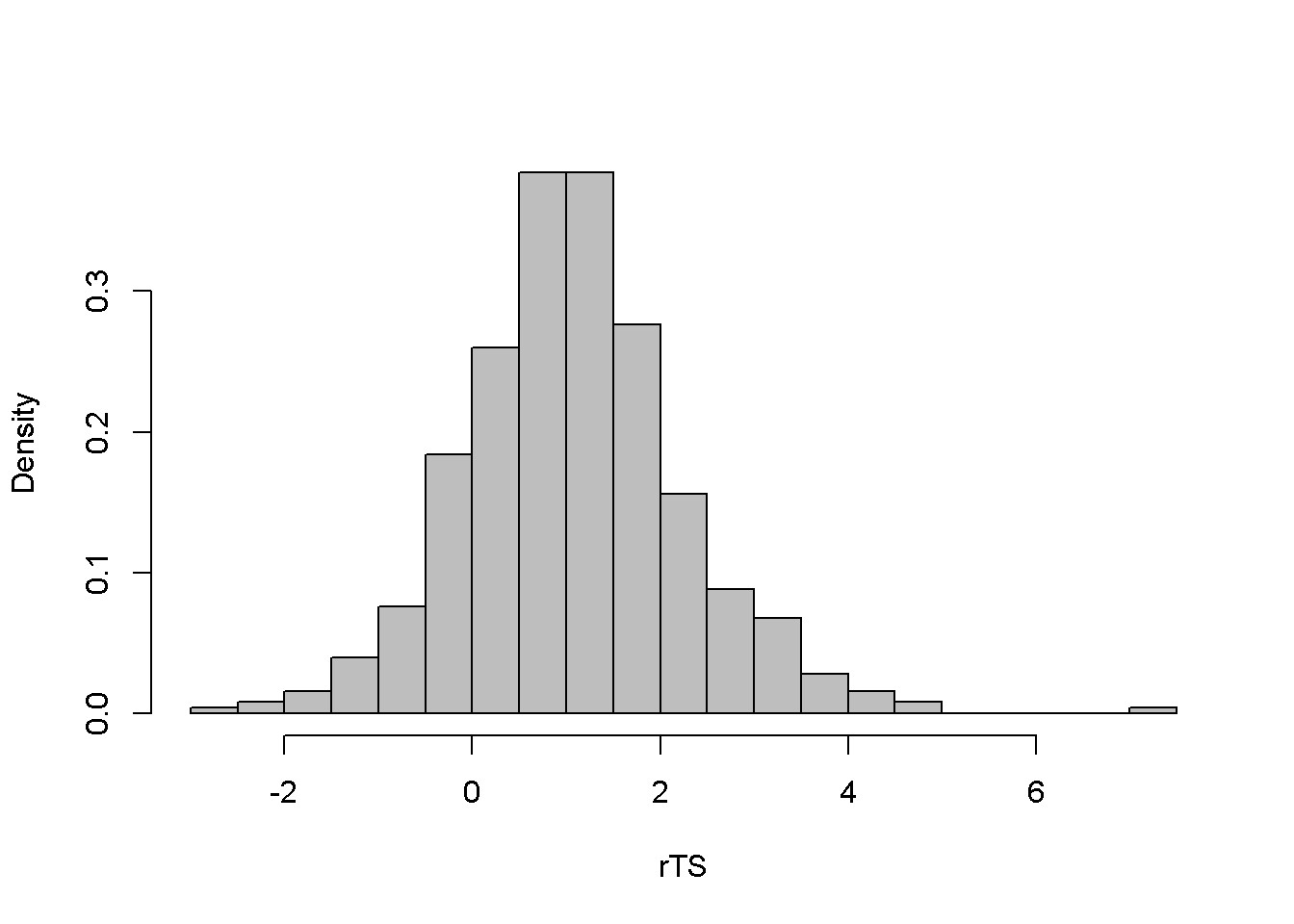


Figure 14.1: Level and histogram of variable ’byd’

byd.mean <- **dynlm**(rTS~1) **summary**(byd.mean)

##

## Time series regression with "ts" data:

## Start = 1, End = 500

##

## Call:

## dynlm(formula = rTS ~ 1)

##

## Residuals:

## Min 1Q Median 3Q Max

## -3.847 -0.723 -0.049 0.669 5.931

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 1.078 0.053 20.4 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 1.19 on 499 degrees of freedom

ehatsq <- **ts**(**resid**(byd.mean)^2) byd.ARCH <- **dynlm**(ehatsq~**L**(ehatsq)) **summary**(byd.ARCH)

##

## Time series regression with "ts" data:

## Start = 2, End = 500

##

## Call:

## dynlm(formula = ehatsq ~ L(ehatsq))

##

## Residuals:

## Min 1Q Median 3Q Max

## -10.802 -0.950 -0.705 0.320 31.347

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## (Intercept) 0.908 0.124 7.30 1.1e-12 \*\*\* ## L(ehatsq) 0.353 0.042 8.41 4.4e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Residual standard error: 2.45 on 497 degrees of freedom

## Multiple R-squared: 0.125, Adjusted R-squared: 0.123

## F-statistic: 70.7 on 1 and 497 DF, p-value: 4.39e-16

|  |
| --- |
| T <- **nobs**(byd.mean) q <- **length**(**coef**(byd.ARCH))-1  Rsq <- **glance**(byd.ARCH)[[1]]  LM <- (T-q)\*Rsq alpha <- 0.05  Chicr <- **qchisq**(1-alpha, q) |

The result is the LM statistic, equal to 62*.*16, which is to be compared to the critical chi-squared value with *α* = 0*.*05 and *q* = 1 degrees of freedom; this value is *χ*2(0*.*95*,*1) = 3*.*84; this indicates that the null hypothesis is rejected, concluding that the series has ARCH effects.

The same conclusion can be reached if, instead of the step-by-step procedure we use one of *R*’s ARCH test capabilities, the ArchTest() function in package FinTS.

bydArchTest <- **ArchTest**(byd, lags=1, demean=TRUE) bydArchTest

##

## ARCH LM-test; Null hypothesis: no ARCH effects

##

## data: byd

## Chi-squared = 62.16, df = 1, p-value = 3.22e-15

Function garch() in the tseries package, becomes an ARCH model when used with the order= argument equal to c(0,1). This function can be used to estimate

*14.1. THE ARCH MODEL*

and plot the variance *ht* defined in Equation 14.3, as shown in the following code and in Figure 14.2.

byd.arch <- **garch**(rTS,**c**(0,1))

##

## \*\*\*\*\* ESTIMATION WITH ANALYTICAL GRADIENT \*\*\*\*\*

##

##

## I INITIAL X(I) D(I)

##

## 1 1.334069e+00 1.000e+00

## 2 5.000000e-02 1.000e+00

##

## IT NF F RELDF PRELDF RELDX STPPAR D\*STEP NPRELDF

## 0 1 5.255e+02

## 1 2 5.087e+02 3.20e-02 7.13e-01 3.1e-01 3.8e+02 1.0e+00 1.34e+02

## 2 3 5.004e+02 1.62e-02 1.78e-02 1.2e-01 1.9e+00 5.0e-01 2.11e-01

## 3 5 4.803e+02 4.03e-02 4.07e-02 1.2e-01 2.1e+00 5.0e-01 1.42e-01

## 4 7 4.795e+02 1.60e-03 1.99e-03 1.3e-02 9.7e+00 5.0e-02 1.36e-02

## 5 8 4.793e+02 4.86e-04 6.54e-04 1.2e-02 2.3e+00 5.0e-02 2.31e-03

## 6 9 4.791e+02 4.16e-04 4.93e-04 1.2e-02 1.7e+00 5.0e-02 1.39e-03 ## 7 10 4.789e+02 3.80e-04 4.95e-04 2.3e-02 4.6e-01 1.0e-01 5.36e-04

## 8 11 4.789e+02 6.55e-06 6.73e-06 9.0e-04 0.0e+00 5.1e-03 6.73e-06

## 9 12 4.789e+02 4.13e-08 3.97e-08 2.2e-04 0.0e+00 9.8e-04 3.97e-08 ## 10 13 4.789e+02 6.67e-11 6.67e-11 9.3e-06 0.0e+00 4.2e-05 6.67e-11

##

## \*\*\*\*\* RELATIVE FUNCTION CONVERGENCE \*\*\*\*\*

##

|  |  |
| --- | --- |
| ## FUNCTION 4.788831e+02 | RELDX 9.327e-06 |
| ## FUNC. EVALS 13 | GRAD. EVALS 11 |
| ## PRELDF 6.671e-11  ## | NPRELDF 6.671e-11 |
| ## I FINAL X(I)  ## | D(I) G(I) |
| ## 1 2.152304e+00 | 1.000e+00 -2.370e-06 |
| ## 2 1.592050e-01 | 1.000e+00 -7.896e-06 |

sbydarch <- **summary**(byd.arch) sbydarch

##

## Call:

## garch(x = rTS, order = c(0, 1))

##

## Model:

## GARCH(0,1)

##

## Residuals:

## Min 1Q Median 3Q Max

## -1.459 0.220 0.668 1.079 4.293

##

## Coefficient(s):

## Estimate Std. Error t value Pr(>|t|)

## a0 2.1523 0.1857 11.59 <2e-16 \*\*\*

## a1 0.1592 0.0674 2.36 0.018 \*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Diagnostic Tests: ## Jarque Bera Test

##

## data: Residuals

## X-squared = 48.57, df = 2, p-value = 2.85e-11

##

##

## Box-Ljung test

##

## data: Squared.Residuals

## X-squared = 0.1224, df = 1, p-value = 0.726

hhat <- **ts**(2\*byd.arch$fitted.values[-1,1]^2) **plot.ts**(hhat)

## 14.2 The GARCH Model

|  |
| --- |
| *# Using package `rugarch` for GARCH models* **library**(rugarch) garchSpec <- **ugarchspec**( variance.model=**list**(model="sGARCH", garchOrder=**c**(1,1)),  mean.model=**list**(armaOrder=**c**(0,0)), distribution.model="std")  garchFit <- **ugarchfit**(spec=garchSpec, data=rTS) **coef**(garchFit) |

*14.2. THE GARCH MODEL*



Figure 14.2: Estimated ARCH(1) variance for the ’byd’ dataset

## mu omega alpha1 beta1 shape

## 1.049280 0.396544 0.495161 0.240827 99.999225

rhat <- garchFit@fit$fitted.values

**plot.ts**(rhat)

hhat <- **ts**(garchFit@fit$sigma^2) **plot.ts**(hhat)

|  |
| --- |
| *# tGARCH*  garchMod <- **ugarchspec**(variance.model=**list**(model="fGARCH",  garchOrder=**c**(1,1), submodel="TGARCH"),  mean.model=**list**(armaOrder=**c**(0,0)), distribution.model="std")  garchFit <- **ugarchfit**(spec=garchMod, data=rTS) **coef**(garchFit) |

## mu omega alpha1 beta1 eta11 shape

## 0.986682 0.352207 0.390636 0.375328 0.339432 99.999884

rhat <- garchFit@fit$fitted.values

**plot.ts**(rhat)

hhat <- **ts**(garchFit@fit$sigma^2) **plot.ts**(hhat)

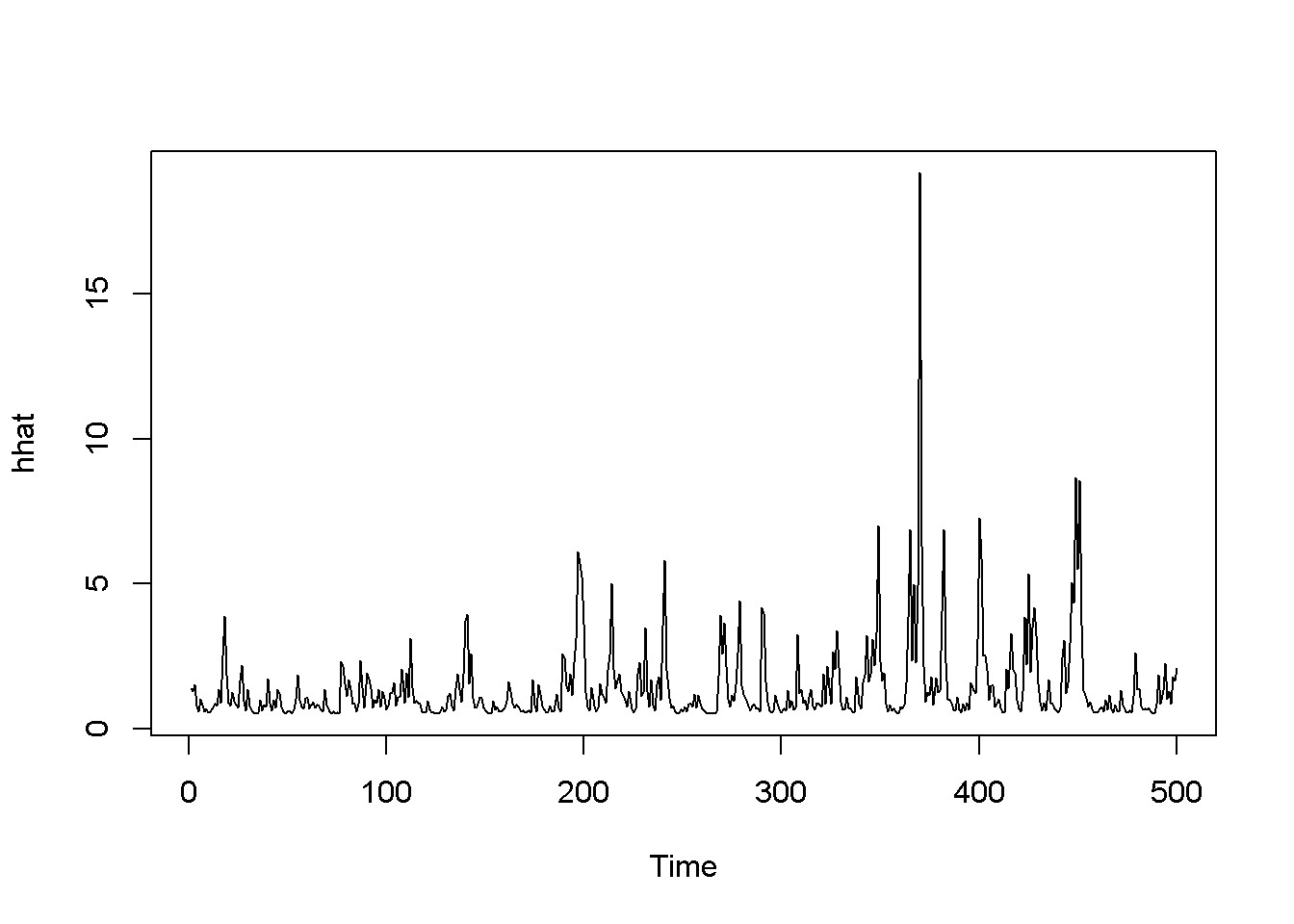
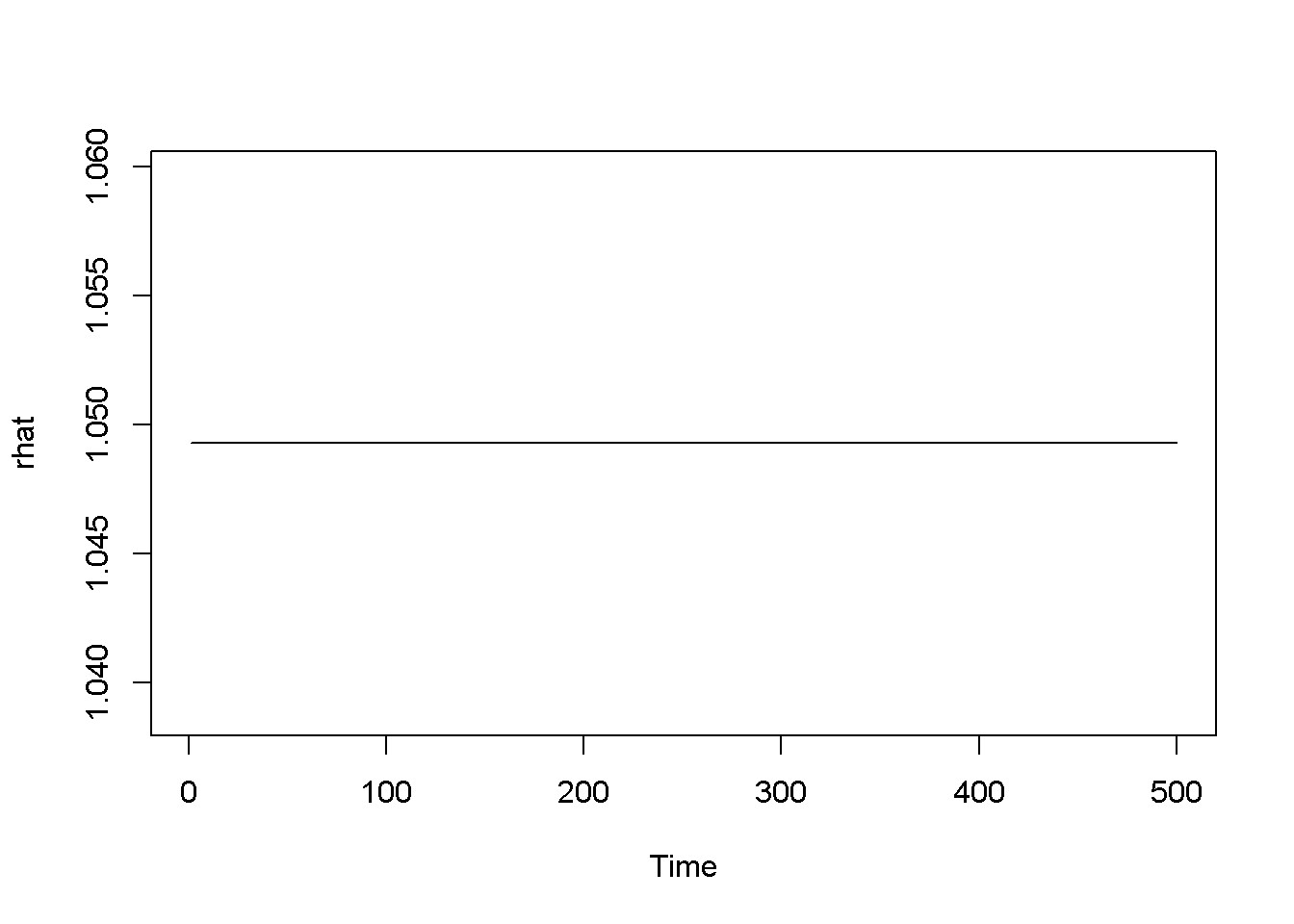


Figure 14.3: Standard GARCH model (sGARCH) with dataset ’byd’

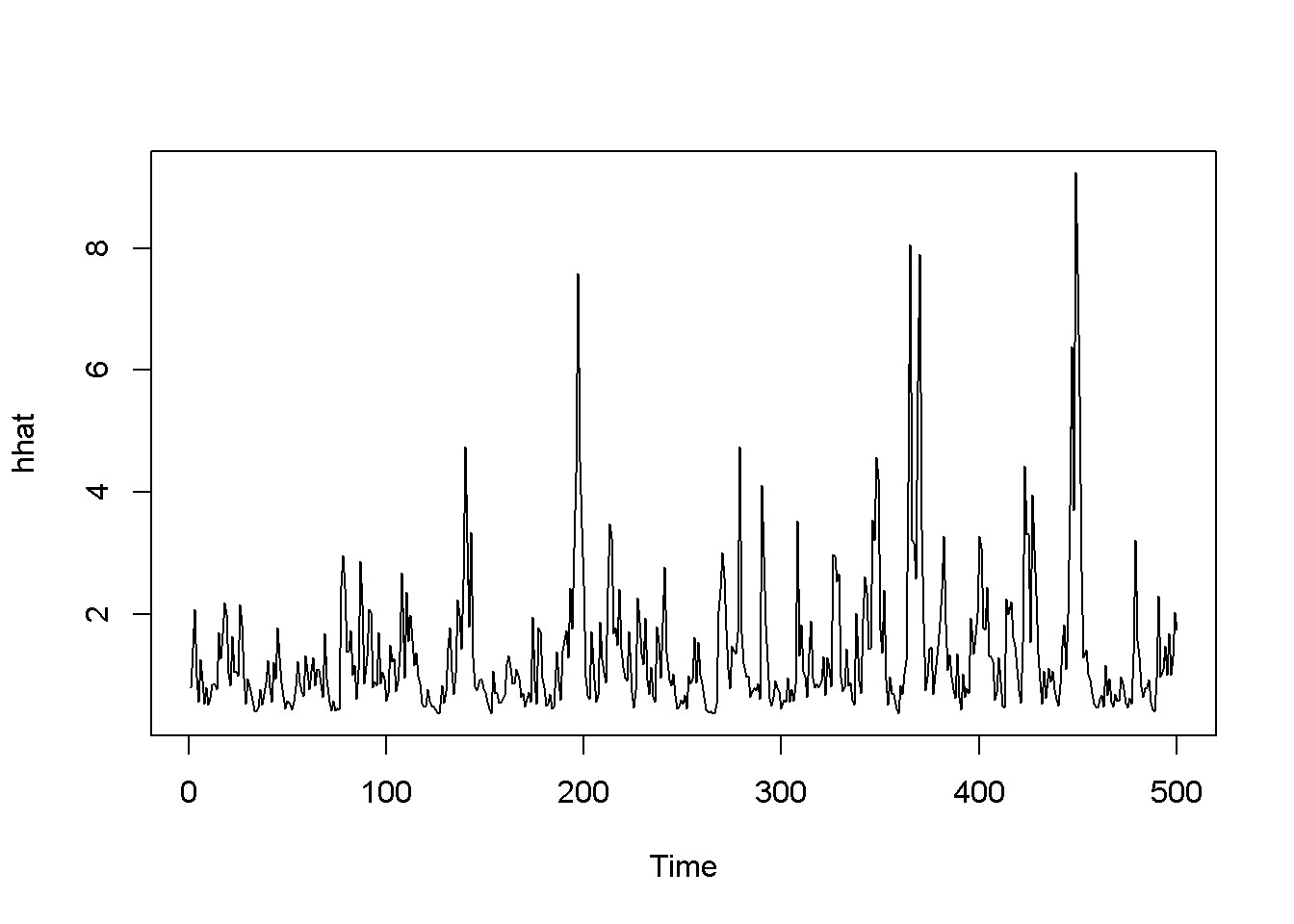
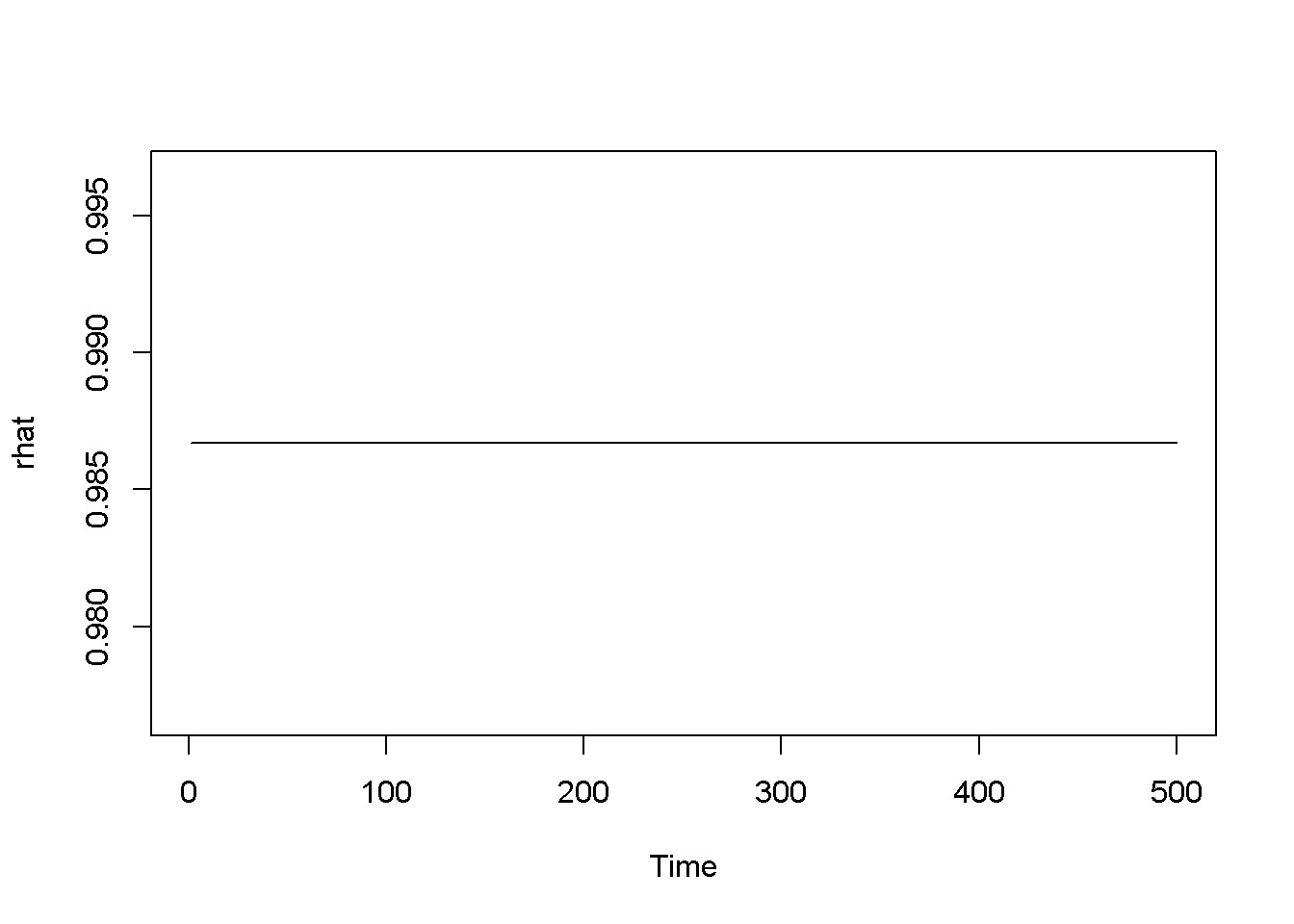


Figure 14.4: The tGARCH model with dataset ’byd’

|  |
| --- |
| *# GARCH-in-mean* garchMod <- **ugarchspec**( variance.model=**list**(model="fGARCH", garchOrder=**c**(1,1), submodel="APARCH"),  mean.model=**list**(armaOrder=**c**(0,0), include.mean=TRUE, archm=TRUE, archpow=2  ),  distribution.model="std"  )  garchFit <- **ugarchfit**(spec=garchMod, data=rTS) **coef**(garchFit) |

*14.2. THE GARCH MODEL*

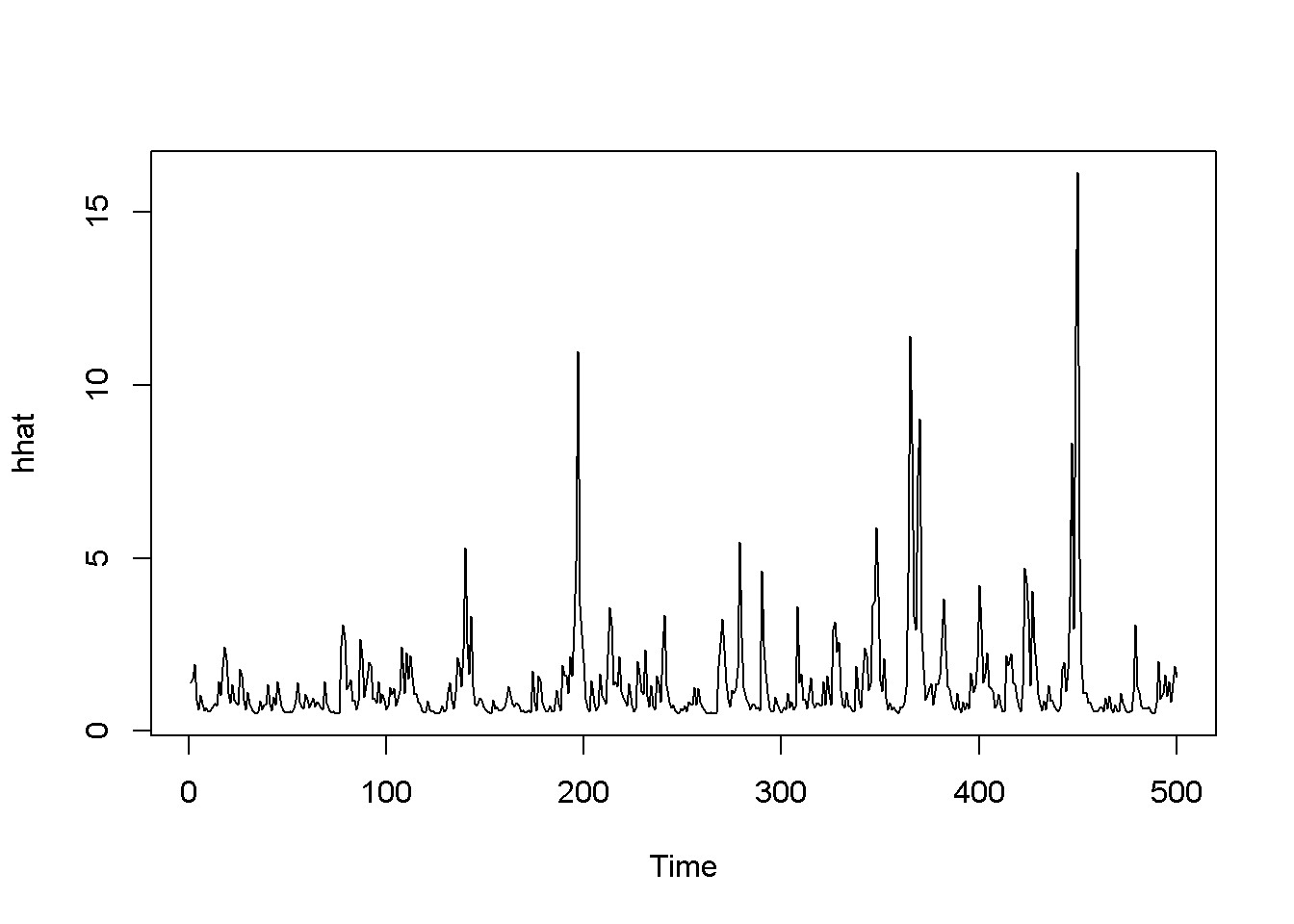
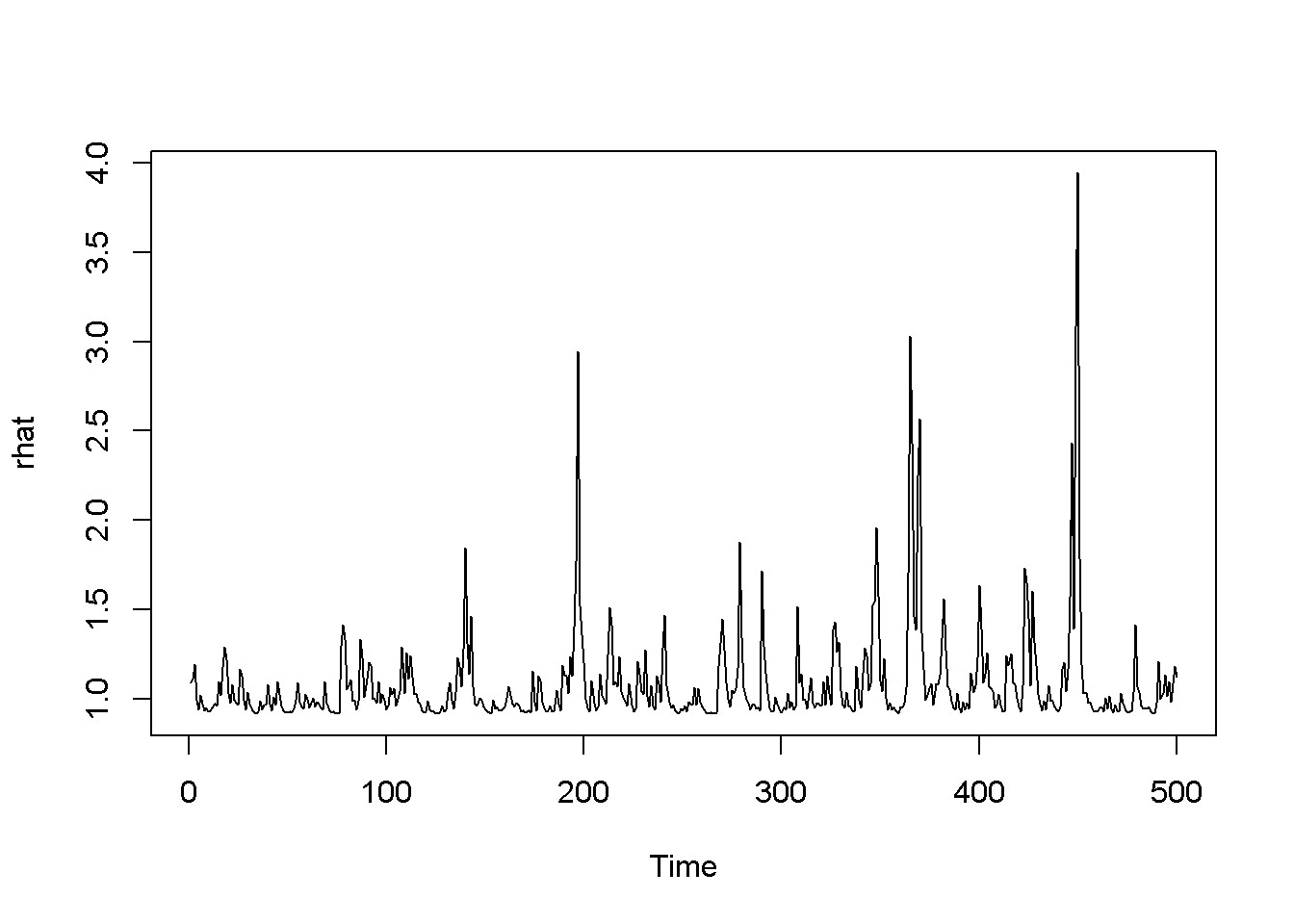


Figure 14.5: A version of the GARCH-in-mean model with dataset ’byd’

## mu archm omega alpha1 beta1 eta11

## 0.820603 0.193476 0.368730 0.442930 0.286748 0.185531

## lambda shape

## 1.897586 100.000000

rhat <- garchFit@fit$fitted.values

**plot.ts**(rhat)

hhat <- **ts**(garchFit@fit$sigma^2) **plot.ts**(hhat)

Figures 14.3, 14.4, and 14.5 show a few versions of the GARCH model. Predictions can be obtained using the function ugarchboot() from the package ugarch.

Optional Lab 2 - Panel Data Models

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!*  **library**(plm)  **library**(tseries) *# for `adf.test()`* **library**(dynlm) *#for function `dynlm()`* **library**(vars) *# for function `VAR()`*  **library**(nlWaldTest) *# for the `nlWaldtest()` function* **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 datasets* **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for `kable()`* **library**(forecast) **library**(systemfit) **library**(AER) **library**(xtable) |

New package: plm (Croissant and Millo 2015).

Panel data gathers information about several individuals (cross-sectional **units**) over several **periods**. The panel is **balanced** if all units are observed in all periods; if some units are missing in some periods, the panel is **unbalanced**. Equation 15.1 gives the form of a pooled panel data model, where the subscript *i* = 1*,...,N* denotes an individual (cross sectional unit), and *t* = 1*,...,T* denotes the time period, or longitudinal unit. The total number of observations in the panel is *N × T*.

*yit* = *β*1 + *β*2*x*2*it* + *...* + *βKxKit* + *eit* (15.1)

217

## 15.1 Organizing the Data as a Panel

A **wide** panel has the cross-sectional dimension (*N*) much larger than the longitudinal dimension (*T*); when the opposite is true, we have a **long** panel. Normally, the same units are observed in all periods; when this is not the case and each period samples mostly other units, the result is not a proper panel data, but **pooled cross-sections** model.

This manual uses the panel data package plm(), which also gives the possibility of organizing the data under the form of a panel. Panel datsets can be organized in mainly two forms: the **long** form has a column for each variable and a row for each individual-period; the **wide** form has a column for each variable-period and a row for each individual. Most panel data methods require the long form, but many data sources provide one wide-form table for each variable; assembling the data from different sources into a long form data frame is often not a trivial matter.

The next code sequence creates a panel structure for the dataset *nls*\_*panel* using the function pdata.frame of the plm package and displays a small part of this dataset.

Please note how the selection of the rows and columns to be displayed is done, using the compact operator %*in*% and arrays such as c(1:6, 14:15). Table 15.1 shows this sample.

|  |
| --- |
| **library**(xtable)  **data**("nls\_panel", package="PoEdata")  nlspd <- **pdata.frame**(nls\_panel, index=**c**("id", "year")) smpl <- nlspd[nlspd$id %in% **c**(1,2),**c**(1:6, 14:15)] tbl <- **xtable**(smpl)  **kable**(tbl, digits=4, align="c", caption="A data sample") |

Function pdim() extracts the dimensions of the panel data:

**pdim**(nlspd)

## Balanced Panel: n=716, T=5, N=3580

## 15.2 The Pooled Model

A **pooled** model has the specification in Equation 15.1, which does not allow for intercept or slope differences among individuals. Such a model can be estimated in *R* using the specification pooling in the plm() function, as the following code sequence illustrates.

*15.2. THE POOLED MODEL*

Table 15.1: A data sample

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | id | year | lwage | hours | age | educ | union | exper |
| 1-82 | 1 | 82 | 1.8083 | 38 | 30 | 12 | 1 | 7.6667 |
| 1-83 | 1 | 83 | 1.8634 | 38 | 31 | 12 | 1 | 8.5833 |
| 1-85 | 1 | 85 | 1.7894 | 38 | 33 | 12 | 1 | 10.1795 |
| 1-87 | 1 | 87 | 1.8465 | 40 | 35 | 12 | 1 | 12.1795 |
| 1-88 | 1 | 88 | 1.8564 | 40 | 37 | 12 | 1 | 13.6218 |
| 2-82 | 2 | 82 | 1.2809 | 48 | 36 | 17 | 0 | 7.5769 |
| 2-83 | 2 | 83 | 1.5159 | 43 | 37 | 17 | 0 | 8.3846 |
| 2-85 | 2 | 85 | 1.9302 | 35 | 39 | 17 | 0 | 10.3846 |
| 2-87 | 2 | 87 | 1.9190 | 42 | 41 | 17 | 1 | 12.0385 |
| 2-88 | 2 | 88 | 2.2010 | 42 | 43 | 17 | 1 | 13.2115 |

Table 15.2: Pooled model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.477 | 0.056 | 8.487 | 0.000 |
| educ | 0.071 | 0.003 | 26.567 | 0.000 |
| exper | 0.056 | 0.009 | 6.470 | 0.000 |
| I(exper^2) | -0.001 | 0.000 | -3.176 | 0.002 |
| tenure | 0.015 | 0.004 | 3.394 | 0.001 |
| I(tenure^2) | 0.000 | 0.000 | -1.886 | 0.059 |
| black | -0.117 | 0.016 | -7.426 | 0.000 |
| south | -0.106 | 0.014 | -7.465 | 0.000 |
| union | 0.132 | 0.015 | 8.839 | 0.000 |

wage.pooled <- **plm**(lwage~educ+exper+**I**(exper^2)+

tenure+**I**(tenure^2)+black+south+union, model="pooling", data=nlspd)

**kable**(**tidy**(wage.pooled), digits=3,

caption="Pooled model")

The plm() function accepts the following main arguments, where the parameters shown as vectors c(...), such as effect and model can only take one value at a time out of the provided list.

plm(formula, data, subset, na.action, effect = c("individual", "time", "twoways"), model = c("within", "random", "ht", "between", "pooling", "fd"),...)

Table 15.3: Pooled ’wage’ model with cluster robust standard errors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.47660 | 0.08441 | 5.64629 | 0.00000 |
| educ | 0.07145 | 0.00549 | 13.01550 | 0.00000 |
| exper | 0.05569 | 0.01129 | 4.93242 | 0.00000 |
| I(exper^2) | -0.00115 | 0.00049 | -2.33440 | 0.01963 |
| tenure | 0.01496 | 0.00711 | 2.10401 | 0.03545 |
| I(tenure^2) | -0.00049 | 0.00041 | -1.18697 | 0.23532 |
| black | -0.11671 | 0.02808 | -4.15602 | 0.00003 |
| south | -0.10600 | 0.02701 | -3.92422 | 0.00009 |
| union | 0.13224 | 0.02703 | 4.89327 | 0.00000 |

tbl <- **tidy**(**coeftest**(wage.pooled, vcov=**vcovHC**(wage.pooled, type="HC0",cluster="group")))

**kable**(tbl, digits=5, caption= "Pooled 'wage' model with cluster robust standard errors")

## 15.3 The Fixed Effects Model

The fixed effects model takes into account individual differences, translated into different intercepts of the regression line for different individuals. The model in this case assigns the subscript *i* to the constant term *β*1, as shown in Equation 15.2; the constant terms calculated in this way are called **fixed effects**.

*yit* = *β*1*i* + *β*2*ix*2*it* + *β*3*ix*3*it* + *eit* (15.2)

Variables that change little or not at all over time, such as some individual characteristics should not be included in a fixed effects model because they produce collinearity with the fixed effects.

nls10 <- **pdata.frame**(nls\_panel[nls\_panel$id %in% 1:10,])

## series nev\_mar, not\_smsa, south are constants and have been removed

|  |
| --- |
| wage.fixed <- **lm**(lwage~exper+**I**(exper^2)+  tenure+**I**(tenure^2)+union+**factor**(id)-1, data=nls10)  **kable**(**tidy**(wage.fixed), digits=3, caption="Fixed effects in a subsample") |

*15.3. THE FIXED EFFECTS MODEL*

Table 15.4: Fixed effects in a subsample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| exper | 0.238 | 0.188 | 1.268 | 0.213 |
| I(exper^2) | -0.008 | 0.008 | -1.036 | 0.307 |
| tenure | -0.012 | 0.034 | -0.362 | 0.720 |
| I(tenure^2) | 0.002 | 0.003 | 0.854 | 0.399 |
| union | 0.114 | 0.151 | 0.753 | 0.457 |
| factor(id)1 | 0.152 | 1.097 | 0.139 | 0.891 |
| factor(id)2 | 0.187 | 1.071 | 0.174 | 0.863 |
| factor(id)3 | -0.063 | 1.351 | -0.047 | 0.963 |
| factor(id)4 | 0.186 | 1.343 | 0.138 | 0.891 |
| factor(id)5 | 0.939 | 1.098 | 0.855 | 0.398 |
| factor(id)6 | 0.794 | 1.112 | 0.715 | 0.480 |
| factor(id)7 | 0.581 | 1.236 | 0.470 | 0.641 |
| factor(id)8 | 0.538 | 1.097 | 0.490 | 0.627 |
| factor(id)9 | 0.418 | 1.084 | 0.386 | 0.702 |
| factor(id)10 | 0.615 | 1.090 | 0.564 | 0.577 |

Table 15.4 displays the results of an OLS regression on a subsample of the first 10 individuals in the dataset *nls*\_*panel*. The table is generated by the previous code sequence, where the novelty is using the factor variable *id*. The function factor() generates dummy variables for all categories of the variable, taking the first category as the reference. To include the reference in the output, one needs to exclude the constant from the regression model by including the term *−*1 in the regression formula. When the constant is not excluded, the coefficients of the dummy variables represent, as usual, the difference between the respective category and the benchmark one.

However, to estimate a fixed effects in *R* we do not need to create the dummy variables, but use the option model="within" in the plm() function. The following code fragment uses the whole sample.

|  |
| --- |
| wage.within <- **plm**(lwage~exper+**I**(exper^2)+ tenure+**I**(tenure^2)+south+union, data=nlspd, model="within")  tbl <- **tidy**(wage.within) **kable**(tbl, digits=5, caption=  "Fixed effects using 'within' with full sample") |

Table 15.5: Fixed effects using ’within’ with full sample

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| exper | 0.04108 | 0.00662 | 6.20590 | 0.00000 |
| I(exper^2) | -0.00041 | 0.00027 | -1.49653 | 0.13463 |
| tenure | 0.01391 | 0.00328 | 4.24333 | 0.00002 |
| I(tenure^2) | -0.00090 | 0.00021 | -4.35357 | 0.00001 |
| south | -0.01632 | 0.03615 | -0.45153 | 0.65164 |
| union | 0.06370 | 0.01425 | 4.46879 | 0.00001 |

Table 15.6: Fixed effects using the ’within’ model option for n=10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| exper | 0.23800 | 0.18776 | 1.26758 | 0.21332 |
| I(exper^2) | -0.00819 | 0.00790 | -1.03584 | 0.30738 |
| tenure | -0.01235 | 0.03414 | -0.36171 | 0.71974 |
| I(tenure^2) | 0.00230 | 0.00269 | 0.85407 | 0.39887 |
| union | 0.11354 | 0.15086 | 0.75263 | 0.45670 |
| wage10.within <- **plm**(lwage~exper+**I**(exper^2)+  tenure+**I**(tenure^2)+union, data=nls10, model="within")  tbl <- **tidy**(wage10.within) **kable**(tbl, digits=5, caption=  "Fixed effects using the 'within' model option for n=10") | | | | | | |

Table 15.6 presents the fixed effects model results for the subsample of 10 individuals of the dataset *nls*\_*panel*. This is to be compared to Table 15.4 to see that the within method is equiivalent to including the dummies in the model. An interesting comparison is between the pooled and fixed effect models. Comparing Table 15.2 with Table 15.5 one can notice that including accounting for individual heterogeneity significantly lowers the marginal effects of the variables.

Testing if fixed effets are necessary is to compare the fixed effects model wage.within with the pooled model wage.pooled. The function pFtest() does this comparison, as in the following code lines.

**kable**(**tidy**(**pFtest**(wage.within, wage.pooled)), caption=

"Fixed effects test: Ho:'No fixed effects'")

Table 15.7 shows that the null hypothesis of no fixed effects is rejected.

*15.4. THE RANDOM EFFECTS MODEL*

Table 15.7: Fixed effects test: Ho:’No fixed effects’

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| df1 | df2 | statistic | p.value | method | Alternative |
| 713 | 2858 | 15.1875 | 0 | F test for individual effects | significant effects |

## 15.4 The Random Effects Model

The **random effects** model elaborates on the fixed effects model by recognizing that, since the individuals in the panel are randomly selected, their characteristics, measured by the intercept *β*1*i* should also be random. Thus, the random effects

model assumes the form of the intercept as given in Equation 15.3, where *β*1 stands for the population average and *ui* represents an individual-specific random term. As in the case of fixed effects, random effects are also time-invariant.

*β*1*i* = *β*1 + *ui* (15.3)

If this form of the intercept is replaced in Equation 15.2, the result looks like Equation 15.4.

*yit* = *β*1 + *β*2*x*2*it* + *νit* (15.4)

The intercept is here, unlike the fixed effects model constant across individuals, but the error termm, *νit*, incorporates both individual specifics and the initial regression error term, as Equation 15.5 shows.

*νit* = *ui* + *eit* (15.5)

Thus, the random effects model is distinguished by the special structure of its error term: errors have zero mean, a variance equal to *σu*2 + *σe*2, uncorrelated across individuals, and having timewise covariance equal to *σu*2.

An important feature of the random effects model is that the timewise correlation in the errors does not decline over time (see Equation 15.6).

*σ*2

*ρ* = *corr*(*νit,νis*) = 2 +*uσe*2 (15.6) *σu*

Testing for random effects amounts to testing the hypothesis that there are no differences among individuals, which implies that the individual-specific random variable has zero variance. Equation 15.7 shows the hypothesis to be tested.

Table 15.8: A random effects test for the wage equation

|  |  |  |  |
| --- | --- | --- | --- |
| statistic | p.value | method | alternative |
| 62.1231 | 0 | Lagrange Multiplier Test - (Honda) | significant effects |

*H*0 : *σu*2 = 0*, HA* : *σu*2 *>* 0 (15.7)

The same function we used for fixed effects can be used for random effects, but setting the argument model= to ‘random’ and selecting the random.method as one out of four possibilities: “swar” (default), “amemiya”, “walhus”, or “nerlove”. The random effects test function is plmtest(), which takes as its main argument the pooling model (indeed it extracts the residuals from the pooling object).

|  |
| --- |
| wageReTest <- **plmtest**(wage.pooled, effect="individual") **kable**(**tidy**(wageReTest), caption=  "A random effects test for the wage equation") |

Table 15.8 shows that the null hypothesis of zero variance in individual-specific errors is rejected; therefore, heterogeneity among individuals may be significant.

Random effects estimator are reliable under the assumption that individual characteristics (heterogeneity) are exogenous, that is, they are independent with respect to the regressors in the random effects equation. The same Hausman test for endogeneity we have already used in another chapter can be used here as well, with the null hypothesis that individual random effects are exogenous. The test function phtest() compares the fixed effects and the random effects models; the next code lines estimate the random effects model and performs the Hausman endogeneity test.

|  |
| --- |
| wage.random <- **plm**(lwage~educ+exper+**I**(exper^2)+ tenure+**I**(tenure^2)+black+south+union, data=nlspd, random.method="swar", model="random")  **kable**(**tidy**(wage.random), digits=4, caption=  "The random effects results for the wage equation") |

**kable**(**tidy**(**phtest**(wage.within, wage.random)), caption=

"Hausman endogeneity test for the random effects wage model")

Table 15.10 shows a low *p*-value of the test, which indicates that the null hypothesis saying that the individual random effects are exogenous is rejected, which makes the random effects equation inconsistent. In this case the fixed effects model is the correct solution. (The number of parameters in Table 15.10 is given for the time-varying variables only.)

*15.4. THE RANDOM EFFECTS MODEL*

Table 15.9: The random effects results for the wage equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.5339 | 0.0799 | 6.6854 | 0.0000 |
| educ | 0.0733 | 0.0053 | 13.7454 | 0.0000 |
| exper | 0.0436 | 0.0064 | 6.8606 | 0.0000 |
| I(exper^2) | -0.0006 | 0.0003 | -2.1363 | 0.0327 |
| tenure | 0.0142 | 0.0032 | 4.4697 | 0.0000 |
| I(tenure^2) | -0.0008 | 0.0002 | -3.8785 | 0.0001 |
| black | -0.1167 | 0.0302 | -3.8652 | 0.0001 |
| south | -0.0818 | 0.0224 | -3.6518 | 0.0003 |
| union | 0.0802 | 0.0132 | 6.0729 | 0.0000 |

Table 15.10: Hausman endogeneity test for the random effects wage model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| statistic | p.value | parameter | method | Alternative |
| 20.745 | 0.002038 | 6 | Hausman Test | one model is inconsistent |

The fixed effects model, however, does not allow time-invariant variables such as *educ* or *black*. Since the problem of the random effects model is endogeneity, one can use instrumental variables methods when time-invariant regressors must be in the model. The **Hausman-Taylor estimator** uses instrumental variables in a random effects model; it assumes four categories of regressors: time-varying exogenous, time-varying endogenous, time-invariant exogenous, and time-invariant endogenous. The number of time-varying variables must be at least equal to the number of time-invariant ones. In our *wage* model, suppose *exper*, *tenure* and *union* are time-varying exogenous, *south* is time-varying endogenous, *black* is time-invariant exogenous, and *educ* is timeinvariant endogenous. The same plm() function allows carrying out Hausman-Taylor estimation by setting model= “ht”.

|  |
| --- |
| wage.HT <- **plm**(lwage~educ+exper+**I**(exper^2)+ tenure+**I**(tenure^2)+black+south+union | exper+**I**(exper^2)+tenure+**I**(tenure^2)+union+black, data=nlspd, model="ht")  **kable**(**tidy**(wage.HT), digits=5, caption=  "Hausman-Taylor estimates for the wage equation") |

Table 15.11 shows the results of the Hausman-Taylor estimation, with the largest changes taking place for *educ* and *black*.

Table 15.11: Hausman-Taylor estimates for the wage equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -0.75077 | 0.58624 | -1.28066 | 0.20031 |
| educ | 0.17051 | 0.04446 | 3.83485 | 0.00013 |
| exper | 0.03991 | 0.00647 | 6.16382 | 0.00000 |
| I(exper^2) | -0.00039 | 0.00027 | -1.46222 | 0.14368 |
| tenure | 0.01433 | 0.00316 | 4.53388 | 0.00001 |
| I(tenure^2) | -0.00085 | 0.00020 | -4.31885 | 0.00002 |
| black | -0.03591 | 0.06007 | -0.59788 | 0.54992 |
| south | -0.03171 | 0.03485 | -0.91003 | 0.36281 |
| union | 0.07197 | 0.01345 | 5.34910 | 0.00000 |

Table 15.12: The head of the grunfeld2 dataset organized as a panel

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | inv | v | k | firm | year |
| 1-1935 | 33.1 | 1170.6 | 97.8 | 1 | 1935 |
| 1-1936 | 45.0 | 2015.8 | 104.4 | 1 | 1936 |
| 1-1937 | 77.2 | 2803.3 | 118.0 | 1 | 1937 |
| 1-1938 | 44.6 | 2039.7 | 156.2 | 1 | 1938 |
| 1-1939 | 48.1 | 2256.2 | 172.6 | 1 | 1939 |
| 1-1940 | 74.4 | 2132.2 | 186.6 | 1 | 1940 |

## 15.5 Grunfeld’s Investment Example

The dataset *grunfeld*2 is a subset of the initial dataset; it includes two firms, GE and WE observed over the period 1935 to 1954. The purpose of this example is to identify various issues that should be taken into account when building a panel data econometric model. The problem is to find the determinants of investment by a firm , *invit* among regressors such as the value of the firm, *vit*, and capital stock *kit*. Table 15.12 gives a glimpse of the grunfeld panel data.

**data**("grunfeld2", package="PoEdata")

grun <- **pdata.frame**(grunfeld2, index=**c**("firm","year"))

**kable**(**head**(grun), align="c", caption= "The head of the grunfeld2 dataset organized as a panel")

Let us consider a pooling model first, assuming that the coefficients of the regression equation, as well as the error variances are the same for both firms (no individual heterogeneity).

Table 15.13: Grunfeld dataset, pooling panel data results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 17.87200 | 7.02408 | 2.54439 | 0.01525 |
| v | 0.01519 | 0.00620 | 2.45191 | 0.01905 |
| k | 0.14358 | 0.01860 | 7.71890 | 0.00000 |

Table 15.14: Grunfeld dataset, ’pooling’ panel data results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -9.9563 | 23.6264 | -0.4214 | 0.6761 |
| v | 0.0266 | 0.0117 | 2.2651 | 0.0300 |
| grun$firm2 | 9.4469 | 28.8054 | 0.3280 | 0.7450 |
| k | 0.1517 | 0.0194 | 7.8369 | 0.0000 |
| v:grun$firm2 | 0.0263 | 0.0344 | 0.7668 | 0.4485 |
| grun$firm2:k | -0.0593 | 0.1169 | -0.5070 | 0.6155 |
| grun.pool <- **plm**(inv~v+k, model="pooling",data=grun)  **kable**(**tidy**(grun.pool), digits=5, caption=  "Grunfeld dataset, pooling panel data results") | | | | | | |

SSE.pool <- **sum**(**resid**(grun.pool)^2) sigma2.pool <- SSE.pool/(grun.pool$df.residual)

For the pooling model, *SSE* = 16563*.*003385, and *σ*2 = 447*.*64874.

Allowing for different coefficients across firms but same error structure is the fixed effects model summarized in Table 15.14. Note that the fixed effects are modeled using the function factor().

|  |
| --- |
| grun.fe <- **plm**(inv~v\*grun$firm+k\*grun$firm, model="pooling",data=grun)  **kable**(**tidy**(grun.fe), digits=4, caption=  "Grunfeld dataset, 'pooling' panel data results") |

SSE.fe <- **sum**(**resid**(grun.fe)^2) sigma2.fe <- SSE.fe/(grun.fe$df.residual)

For the fixed effects model with firm dummies, *SSE* = 14989*.*821701, and *σ*2 = 440*.*877109.

A test to see if the coefficients are significantly different between the pooling and fixed Table 15.15: Pooling astimates for the GE firm (firm=1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -9.9563 | 31.3742 | -0.3173 | 0.7548 |
| v | 0.0266 | 0.0156 | 1.7057 | 0.1063 |
| k | 0.1517 | 0.0257 | 5.9015 | 0.0000 |

effects equations can be done in *R* using the function pooltest from package plm; to perform this test, the fixed effects model should be estimated with the function pvcm with the argument model= “within”, as the next code lines show.

grun.pvcm <- **pvcm**(inv~v+k, model="within", data=grun)

**coef**(grun.pvcm)

|  |  |  |
| --- | --- | --- |
| (Intercept) | v | K  0.151694  0.092406 |
| -9.95631 | 0.026551 |
| -0.50939 | 0.052894 |
| **pooltest**(grun.pool, grun.pvcm) | | |

##

## F statistic

##

## data: inv ~ v + k

## F = 1.189, df1 = 3, df2 = 34, p-value = 0.328

## alternative hypothesis: unstability

The result shows that the null hypothesis of zero coefficients for the individual dummy terms are zero cannot be rejected. (However, the pvcm function is not equivalent to the fixed effects model that uses individual dummies; it is, though, useful for testing the ‘poolability’ of a dataset.)

Now, if we allow for different coefficients *and* different error variances, the equations for each individual is independent from those for other individuals and it can be estimated separately.

grun1.pool <- **plm**(inv~v+k, model="pooling", subset=grun$firm==1, data=grun)

SSE.pool1<- **sum**(**resid**(grun1.pool)^2) sig2.pool1 <- SSE.pool1/grun1.pool$df.residual **kable**(**tidy**(grun1.pool), digits=4, align='c', caption=

"Pooling astimates for the GE firm (firm=1)")

Table 15.16: Pooling estimates for the WE firm (firm=2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -0.5094 | 8.0153 | -0.0636 | 0.9501 |
| v | 0.0529 | 0.0157 | 3.3677 | 0.0037 |
| k | 0.0924 | 0.0561 | 1.6472 | 0.1179 |

grun2.pool <- **plm**(inv~v+k, model="pooling", subset=grun$firm==2, data=grun)

SSE.pool2 <- **sum**(**resid**(grun2.pool)^2) sig2.pool2 <- SSE.pool2/grun2.pool$df.residual **kable**(**tidy**(grun2.pool), digits=4, align='c', caption=

"Pooling estimates for the WE firm (firm=2)")

Tables 15.15 and 15.16 show the results for the equations on subsets of data, separated by firms.

A Godfeld-Quandt test can be carried out to determine whether the variances are different among firms, as the next code shows.

**gqtest**(grun.pool, point=0.5, alternative="two.sided", order.by=grun$firm)

##

## Goldfeld-Quandt test

##

## data: grun.pool

## GQ = 0.1342, df1 = 17, df2 = 17, p-value = 0.000143

The result is rejection of the null hypothesis that the variances are equal, indicating that estimating separate equations for each firm is the correct model.

What happens when we assume that the only link between the two firms is correlation between their contemporaneous error terms? This is the model of **seemingly unrelated regressions**, a generalized least squares method.

|  |
| --- |
| **library**("systemfit") grunf<- grunfeld2 grunf$Firm<-"WE" for (i in 1:40){ if(grunf$firm[i]==1){grunf$Firm[i] <- "GE"}  }  grunf$firm <- NULL  **names**(grunf)<- **c**("inv", "val", "cap", "year", "firm") |

grunfpd <- **plm.data**(grunf, **c**("firm","year"))

grunf.SUR <- **systemfit**(inv~val+cap, method="SUR", data=grunfpd) **summary**(grunf.SUR, resdCov=FALSE, equations=FALSE)

##

## systemfit results

## method: SUR

##

## N DF SSR detRCov OLS-R2 McElroy-R2 ## system 40 34 15590 35641 0.69897 0.6151

##

## N DF SSR MSE RMSE R2 Adj R2

## GE 20 17 13788.4 811.08 28.479 0.69256 0.65639 ## WE 20 17 1801.3 105.96 10.294 0.74040 0.70986

##

## The covariance matrix of the residuals used for estimation

## GE WE ## GE 777.45 207.59

## WE 207.59 104.31

##

## The covariance matrix of the residuals

## GE WE ## GE 811.08 224.28

## WE 224.28 105.96

##

## The correlations of the residuals

## GE WE ## GE 1.00000 0.76504

## WE 0.76504 1.00000

##

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

## GE\_(Intercept) -27.719317 29.321219 -0.9454 0.357716

## GE\_val 0.038310 0.014415 2.6576 0.016575 \*

## GE\_cap 0.139036 0.024986 5.5647 0.00003423 \*\*\*

## WE\_(Intercept) -1.251988 7.545217 -0.1659 0.870168

## WE\_val 0.057630 0.014546 3.9618 0.001007 \*\*

## WE\_cap 0.063978 0.053041 1.2062 0.244256

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 First, please note that the systemfit() function requires a panel data file created with plm.data, instead of the pdata.frame that we have used above; second, for some reason I had to change the names of the variables to names having more than one letter to make the function work. I did this using the function names().

232 *CHAPTER 15. PANEL DATA MODELS*

**Chapter 16**

# Qualitative and LDV Models

|  |
| --- |
| **rm**(list=**ls**()) *#Removes all items in Environment!*  **library**(nlWaldTest) *# for the `nlWaldtest()` function* **library**(lmtest) *#for `coeftest()` and `bptest()`.*  **library**(broom) *#for `glance(`) and `tidy()`* **library**(PoEdata) *#for PoE4 datasets* **library**(car) *#for `hccm()` robust standard errors* **library**(sandwich)  **library**(knitr) *#for `kable()`*  **library**(forecast) **library**(AER) **library**(xtable) |

## 16.1 The Linear Probability Model

Suppose the response variable is binary, as defined in Equation 16.1.

*y* = 1 if an individual chooses to buy a house*y* = 0 if an individual chooses not to buy

(16.1)

The linear probability model has the general form is shown in Equation 16.2. *E*(*y*) is the probability that the response variable takes the value of 1; therefore, a predicted value of *y* is a prediction for the probability that *y* = 1.

*y* = *E*(*y*) + *e* = *β*1 + *β*2*x*2 + *...* + *βkxk* + *e* (16.2)

233

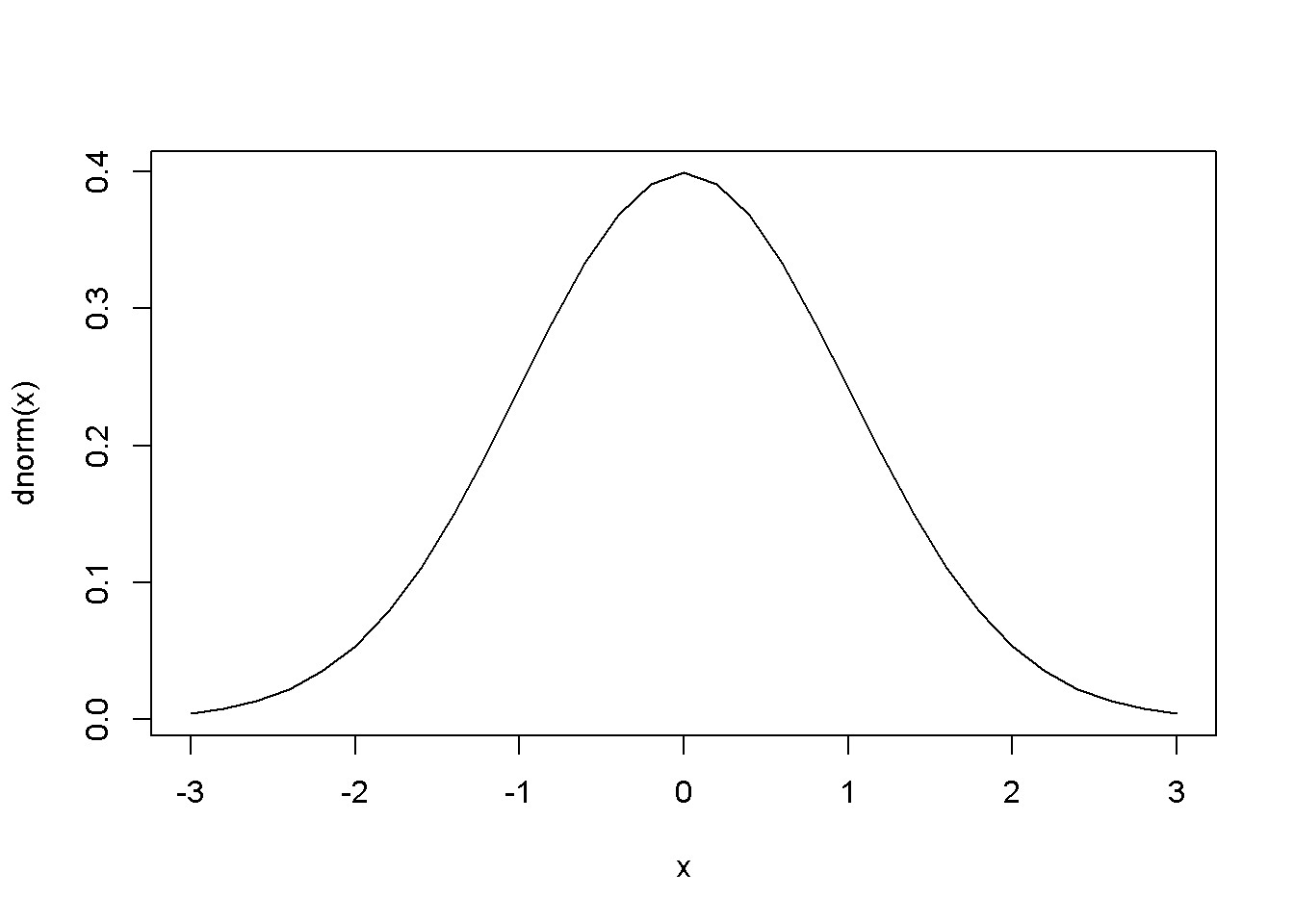
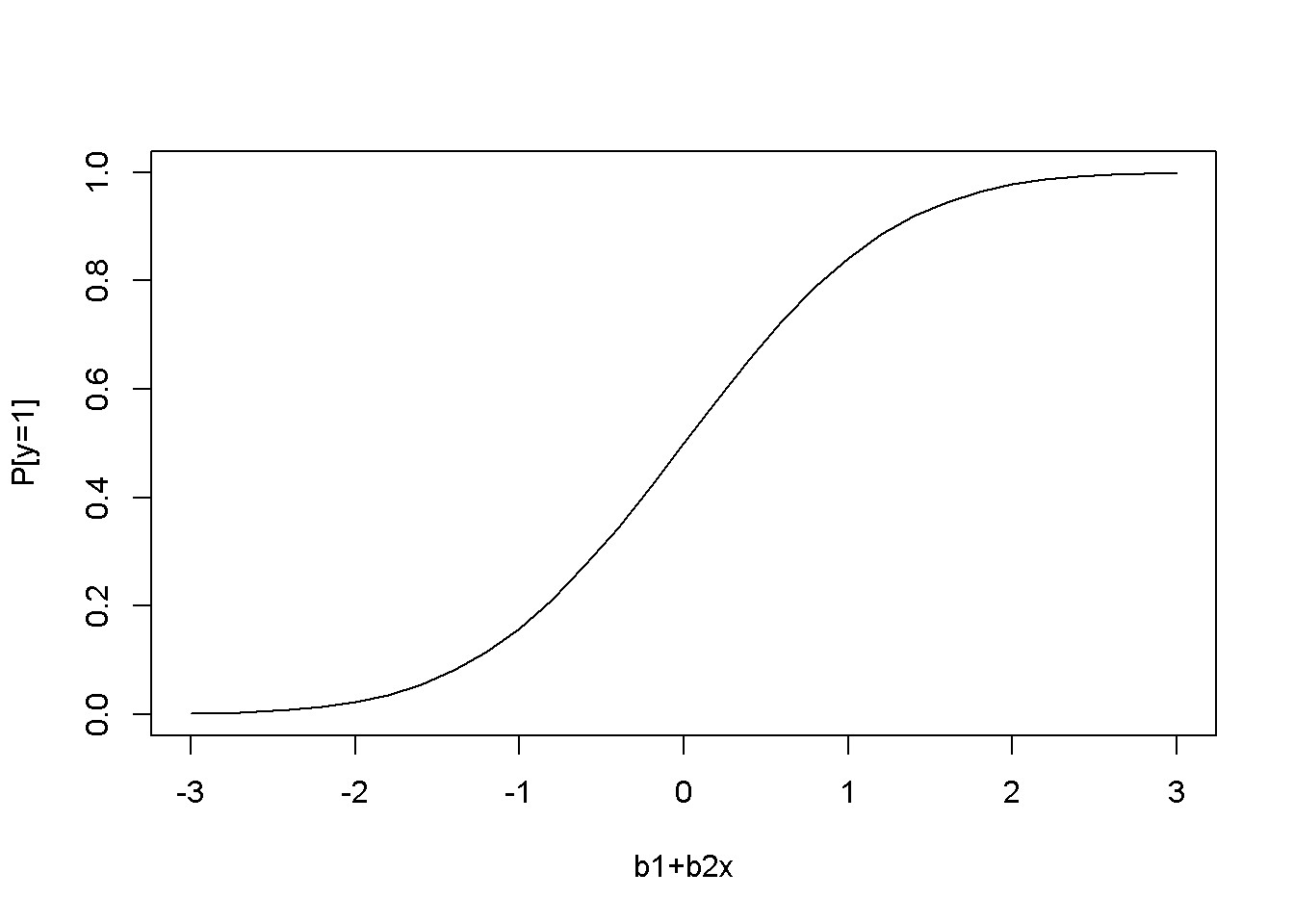


Figure 16.1: The shape of the probit function is the standard normal distribution

## 16.2 The Probit Model

The **probit** model assumes a nonlinear relationship between the response variable and regressors, this relationship being the cumulative distribution function of the normal distribution (see Equation 16.3 and Figure 16.1, left).

*p* = *P*[*y* = 1] = *E*(*y|x*) = Φ(*β*1 + *β*2*x*) (16.3)

The slope of the regression curve is not constant, but is given by the standard normal density function (Figure 16.1, right); the slope can be calculated using Equation

16.4.

*dp*

= *φ*(*β*1 + *β*2*x*)*β*2 (16.4) *dx*

Predictions of the probability that *y* = 1 are given by Equation 16.5.

*p*ˆ = Φ(*β*ˆ1 + *β*ˆ2*x*) (16.5)

x <- **seq**(-3,3, .2)

**plot**(x, **pnorm**(x), type="l", xlab="b1+b2x", ylab="P[y=1]") **plot**(x, **dnorm**(x), type="l")

## 16.3 The Transportation Example

The dataset *transport* containes *N* = 21 observations of transportation chioces (*auto* = 1 if individual *i* chooses *auto* and 0 if individual *i* chooses *bus*). The

*16.3. THE TRANSPORTATION EXAMPLE*

Table 16.1: Transport example, estimated by probit

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -0.0644 | 0.4007 | -0.1608 | 0.8722 |
| dtime | 0.3000 | 0.1029 | 2.9154 | 0.0036 |

choice depends on the difference in time between the two means of transportation, *dtime* = (*bustime − autotime*) *÷* 10.

The *R* function to estimate a probit model is glm, with the family argument equal to binomial(link="probit"). The glm function has the following general structure:

glm(formula, family, data, ...)

**data**("transport", package="PoEdata")

auto.probit <- **glm**(auto~dtime, family=**binomial**(link="probit"), data=transport)

**kable**(**tidy**(auto.probit), digits=4, align='c', caption= "Transport example, estimated by probit")

Equation 16.4 can be used to calculate partial effects of an increase in *dtime* by one unit (10 minutes). The following code lines calculate this effect at *dtime* = 2 (time difference of 20 minutes).

xdtime <- **data.frame**(dtime=2) predLinear <- **predict**(auto.probit, xdtime, data=transport, type="link")

DpDdtime <- **coef**(auto.probit)[[2]]\***dnorm**(predLinear)

DpDdtime

## 1

## 0.10369

Predictions can be calculated using the function predict, which has the following general form:

predict(object, newdata = NULL, type = c(“link”, “response”, “terms”), se.fit = FALSE, dispersion = NULL, terms = NULL, na.action = na.pass,

...)

The optional argument newdata must be a data frame containing the new values of the regressors for which the prediction is desired; if missing, prediction is calculated for all observations in the sample.

Here is how to calculate the predicted probability of choosing *auto* when the time difference is 30 minutes (*dtime* = 3):

xdtime <- **data.frame**(dtime=3) predProbit <- **predict**(auto.probit, xdtime, data=transport, type="response")

predProbit

## 1

## 0.798292

The marginal effect at the average predicted value can be determined as follows:

|  |
| --- |
| avgPredLinear <- **predict**(auto.probit, type="link")  avgPred <- **mean**(**dnorm**(avgPredLinear)) AME <- avgPred\***coef**(auto.probit) AME |

## (Intercept) dtime

## -0.0103973 0.0484069

## 16.4 The Logit Model for Binary Choice

This is very similar to the *probit* model, with the difference that **logit** uses the logistic function Λ to link the linear expression *β*1 + *β*2*x* to the probability that the response variable is equal to 1. Equations 16.6 and 16.7 give the defining expressions of the *logit* model (the two expressions are equivalent).

1

*p* = Λ(*β*1 + *β*2*x*) = 1 + *e−*(*β*1+*β*2*x*) (16.6)

*exp*(*β*1 + *β*2*x*)

*p* = (16.7)

1 + *exp*(*β*1 + *β*2*x*)

Equation 16.8 gives the marginal effect of a change in the regressor *xk* on the probability that *y* = 1.

*∂p*

= *βk*Λ(1 *−* Λ) (16.8) *∂xk*

**data**("coke", package="PoEdata")

coke.logit <- **glm**(coke~pratio+disp\_coke+disp\_pepsi,

data=coke, family=**binomial**(link="logit"))

**kable**(**tidy**(coke.logit), digits=5, align="c", caption="Logit estimates for the 'coke' dataset")

*16.4. THE LOGIT MODEL FOR BINARY CHOICE*

Table 16.2: Logit estimates for the ’coke’ dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 1.92297 | 0.32582 | 5.90200 | 0.00000 |
| pratio | -1.99574 | 0.31457 | -6.34437 | 0.00000 |
| disp\_coke | 0.35160 | 0.15853 | 2.21781 | 0.02657 |
| disp\_pepsi | -0.73099 | 0.16783 | -4.35551 | 0.00001 |
| coke.LPM <- **lm**(coke~pratio+disp\_coke+disp\_pepsi, data=coke)  coke.probit <- **glm**(coke~pratio+disp\_coke+disp\_pepsi,  data=coke, family=**binomial**(link="probit"))  **stargazer**(coke.LPM, coke.probit, coke.logit, header=FALSE,  title="Three binary choice models for the 'coke' dataset", type=.stargazertype,  keep.stat="n",digits=4, single.row=FALSE, intercept.bottom=FALSE, model.names=FALSE,  column.labels=**c**("LPM","probit","logit")) | | | | | | |

Prediction and marginal effects for the *logit* model can be determined using the same predict function as for the *probit* model, and Equation 16.8 for marginal effects.

|  |
| --- |
| tble <- **data.frame**(**table**(true=coke$coke,  predicted=**round**(**fitted**(coke.logit))))  **kable**(tble, align='c', caption="Logit prediction results") |

A useful measure of the predictive capability of a binary model is the number of cases correctly predicted. The following table (created by the above code lines) gives these numbers separated by the boinary choice values; the numbers have been determined by rounding the predicted probabilities from the *logit* model.

The usual functions for hypothesis testing, such as anova, coeftest, waldtest and linear.hypothesis are available for these models.

Hnull <- "disp\_coke+disp\_pepsi=0" **linearHypothesis**(coke.logit, Hnull)

|  |  |  |  |
| --- | --- | --- | --- |
| Res.Df | Df | Chisq | Pr(>Chisq) |
| 1137 | NA | NA | NA |
| 1136 | 1 | 5.61053 | 0.017853 |

The above code tests the hypothesis that the effects of displaying coke and displaying Table 16.3: Three binary choice models for the ’coke’ dataset

*Dependent variable:*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | coke |  |
|  | LPM | probit | logit |
| pratio | *−*0.4009*∗∗∗* | *−*1.1459*∗∗∗* | *−*1.9957*∗∗∗* |
|  | (0.0613) | (0.1839) | (0.3146) |
| disp\_coke | 0.0772*∗∗* | 0.2172*∗∗* | 0.3516*∗∗* |
|  | (0.0344) | (0.0962) | (0.1585) |
| disp\_pepsi | *−*0.1657*∗∗∗* | *−*0.4473*∗∗∗* | *−*0.7310*∗∗∗* |
|  | (0.0356) | (0.1010) | (0.1678) |
| Observations | 1,140 | 1,140 | 1,140 |

*Note: ∗*p*<*0.1; *∗∗*p*<*0.05; *∗∗∗*p*<*0.01

Table 16.4: Logit prediction results

|  |  |  |
| --- | --- | --- |
| true | predicted | Freq |
| 0 | 0 | 507 |
| 1 | 0 | 263 |
| 0 | 1 | 123 |
| 1 | 1 | 247 |

*16.5. MULTINOMIAL LOGIT*

pepsi have equal but opposite effects, a null hypothesis that is being rejected by the test. Here is another example, testing the null hypothesis that displaying coke and pepsi have (jointly) no effect on an individual’s choice. This hypothesis is also rejected.

Hnull <- **c**("disp\_coke=0", "disp\_pepsi=0") **linearHypothesis**(coke.logit, Hnull)

|  |  |  |  |
| --- | --- | --- | --- |
| Res.Df | Df | Chisq | Pr(>Chisq) |
| 1138 | NA | NA | NA |
| 1136 | 2 | 18.9732 | 0.000076 |

## 16.5 Multinomial Logit

A relatively common *R* function that fits multinomial logit models is multinom from package nnet. Let us use the dataset *nels\_small* for an example of how multinom works. The variable *grades* in this dataset is an index, with best grades represented by lower values of *grade*. We try to explain the choice of a secondary institution (*psechoice*) only by the high school grade. The variable *pschoice* can take one of three values:

* *psechoice* = 1 no college,
* *psechoice* = 2 two year college
* *psechoice* = 3 four year college

**library**(nnet)

**data**("nels\_small", package="PoEdata")

nels.multinom <- **multinom**(psechoice~grades, data=nels\_small)

## # weights: 9 (4 variable) ## initial value 1098.612289

## iter 10 value 875.313116

## final value 875.313099

## converged

**summary**(nels.multinom)

## Call:

## multinom(formula = psechoice ~ grades, data = nels\_small)

##

## Coefficients:

## (Intercept) grades

## 2 2.50527 -0.308640

## 3 5.77017 -0.706247

##

## Std. Errors:

## (Intercept) grades ## 2 0.418394 0.0522853

## 3 0.404329 0.0529264

##

## Residual Deviance: 1750.63

## AIC: 1758.63

The output from function multinom gives coefficient estimates for each level of the response variable *psechoice*, except for the first level, which is the benchmark.

medGrades <- **median**(nels\_small$grades)

fifthPercentileGrades <- **quantile**(nels\_small$grades, .05) newdat <- **data.frame**(grades=**c**(medGrades, fifthPercentileGrades))

pred <- **predict**(nels.multinom, newdat, "probs") pred

|  |  |  |  |
| --- | --- | --- | --- |
|  | 1 | 2 | 3 |
|  | 0.181018 | 0.285573 | 0.533409 |
| 5% | 0.017818 | 0.096622 | 0.885560 |

The above code lines show how the usual function predict can calculate the predicted probabilities of choosing any of the three secondary education levels for two arbitrary grades: one at the median grade in the sample, and the other at the top fifth percent.

## 16.6 The Conditional Logit Model

In the multinomial logit model all individuals faced the same external conditions and each individual’s choice is only determined by an individual’s circumstances or preferences. The **conditional logit model** allows for individuals to face individualspecific external conditions, such as the price of a product.

Suppose we want to study the effect of price on an individual’s decision about choosing one of three brands of soft drinks:

1. pepsi
2. sevenup
3. coke

In the conditional logit model, the probability that individual *i* chooses brand *j* is given by Equation 16.9.

*16.6. THE CONDITIONAL LOGIT MODEL*

*exp*(*β*1*j* + *β*2*priceij*)

*pij* = (16.9) *exp*(*β*11 + *β*2*pricei*1) + *exp*(*β*12 + *β*2*pricei*2) + *exp*(*β*13 + *β*2*pricei*3)

In Equation 16.9 not all parameters *β*11, *β*12, and *β*13 can be estimated, and therefore one will be set equal to zero. Unlike in the multinomial logit model, the coefficient on the independent variable *price* is the same for all choices, but the value of the independent variable is different for each individual.

*R* offers several alternatives that allow fitting conditional logit models, one of which is the function MCMCmnl() from the package MCMCpack (others are, for instance, clogit() in the survival package and mclogit() in the mclogit package). The following code is adapted from (Adkins 2014).

|  |
| --- |
| **library**(MCMCpack)  **data**("cola", package="PoEdata")  N <- **nrow**(cola) N3 <- N/3  price1 <- cola$price[**seq**(1,N,by=3)] price2 <- cola$price[**seq**(2,N,by=3)] price3 <- cola$price[**seq**(3,N,by=3)]  bchoice <- **rep**("1", N3) for (j in 1:N3){ if(cola$choice[3\*j-1]==1) bchoice[j] <- "2" if(cola$choice[3\*j]==1) bchoice[j] <- "3"  } cola.clogit <- **MCMCmnl**(bchoice ~ **choicevar**(price1, "b2", "1")+ **choicevar**(price2, "b2", "2")+ **choicevar**(price3, "b2", "3"), baseline="3", mcmc.method="IndMH") |

## Calculating MLEs and large sample var-cov matrix.

## This may take a moment...

## Inverting Hessian to get large sample var-cov matrix.

|  |
| --- |
| sclogit <- **summary**(cola.clogit)  tabMCMC <- **as.data.frame**(sclogit$statistics)[,1:2] **row.names**(tabMCMC)<- **c**("b2","b11","b12") **kable**(tabMCMC, digits=4, align="c",  caption="Conditional logit estimates for the 'cola' problem") |

Table 16.5 shows the estimated parameters *βij* in Equation 16.9, with choice 3 (coke) Table 16.5: Conditional logit estimates for the ’cola’ problem

|  |  |  |
| --- | --- | --- |
|  | Mean | SD |
| b2 | -2.2991 | 0.1382 |
| b11 | 0.2839 | 0.0610 |
| b12 | 0.1037 | 0.0621 |

being the baseline, which makes *β*13 equal to zero. Using the *β*s in Table 16.5, let us calculate the probability that individual *i* chooses *pepsi* and *sevenup* for some given values of the prices that individual *i* faces. The calculations follow the formula in Equation 16.9, with *β*13 = 0. Of course, the probability of choosing the baseline brand, in this case Coke, must be such that the sum of all three probabilities is equal to 1.

|  |
| --- |
| pPepsi <- 1 pSevenup <- 1.25 pCoke <- 1.10 b13 <- 0  b2 <- tabMCMC$Mean[1] b11 <- tabMCMC$Mean[2] b12 <- tabMCMC$Mean[3]  *# The probability that individual i chooses Pepsi:*  PiPepsi <- **exp**(b11+b2\*pPepsi)/  (**exp**(b11+b2\*pPepsi)+**exp**(b12+b2\*pSevenup)+ **exp**(b13+b2\*pCoke))  *# The probability that individual i chooses Sevenup:*  PiSevenup <- **exp**(b12+b2\*pSevenup)/  (**exp**(b11+b2\*pPepsi)+**exp**(b12+b2\*pSevenup)+ **exp**(b13+b2\*pCoke))  *# The probability that individual i chooses Coke:*  PiCoke <- 1-PiPepsi-PiSevenup |

The calculatred probabilities are:

* *pi,pepsi* = 0*.*483
* *pi,sevenup* = 0*.*227
* *pi,coke* = 0*.*289

The three probabilities are different for different individuals because different individuals face different prices; in a more complex model other regressors may be included, some of which may reflect individual characteristics.

*16.7. ORDERED CHOICE MODELS*

Table 16.6: Ordered probit estimates for the ’nels’ problem

|  |  |  |
| --- | --- | --- |
|  | Mean | SD |
| (Intercept) | 2.9542 | 0.1478 |
| grades | -0.3074 | 0.0193 |
| gamma2 | 0.8616 | 0.0487 |

## 16.7 Ordered Choice Models

The order of choices in these models is meaningful, unlike the multinomial and conditional logit model we have studied so far. The following example explains the choice of higher education, when the choice variable is *psechoice* and the only regressor is *grades*; the dataset, *nels*\_*small*, is already known to us.

The *R* package MCMCpack is again used here, with its function MCMCoprobit().

|  |
| --- |
| **library**(MCMCpack)  nels.oprobit <- **MCMCoprobit**(psechoice ~ grades, data=nels\_small, mcmc=10000)  sOprobit <- **summary**(nels.oprobit) tabOprobit <- sOprobit$statistics[, 1:2] **kable**(tabOprobit, digits=4, align="c",  caption="Ordered probit estimates for the 'nels' problem") |

Table 16.6 gives the ordered probit estimates. The results from MCMCoprobit can be translated into the textbook notations as follows:

* *µ*1 = *−*(Intercept)
* *β* = grades
* *µ*2 = gamma2 *−* (Intercept)

The probabilities for each choice can be calculated as in the next code fragment:

|  |
| --- |
| mu1 <- -tabOprobit[1] b <- tabOprobit[2] mu2 <- tabOprobit[3]-tabOprobit[1] xGrade <- **c**(**mean**(nels\_small$grades), **quantile**(nels\_small$grades, 0.05))  *# Probabilities:* prob1 <- **pnorm**(mu1-b\*xGrade)  prob2 <- **pnorm**(mu2-b\*xGrade)-**pnorm**(mu1-b\*xGrade) prob3 <- 1-**pnorm**(mu2-b\*xGrade) |

Table 16.7: Poisson model for the ’olympics’ problem

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -16.0767 | 0.1732 | -92.8143 | 0 |
| log(pop) | 0.2080 | 0.0118 | 17.6419 | 0 |
| log(gdp) | 0.5701 | 0.0087 | 65.5780 | 0 |

*# Marginal effects:*

Dp1DGrades <- -**pnorm**(mu1-b\*xGrade)\*b

Dp2DGrades <- (**pnorm**(mu1-b\*xGrade)-**pnorm**(mu2-b\*xGrade))\*b

Dp3DGrades <- **pnorm**(mu2-b\*xGrade)\*b

For instance, the marginal effect of *grades* on the probability of attending a four-year college for a student with average grade and for a student in the top 5 percent are, respectively, *−*0*.*143 and *−*0*.*031.

## 16.8 Models for Count Data

Such models use the Poisson distribution function, of the (count) variable *y*, as shown in Equations 16.10 and 16.11.

*e−λλy*

*f*(*y*) = *P*(*Y* = *y*) = (16.10) *y*!

*E*(*y*) = *λ* = *exp*(*β*1 + *β*2*x*)*y* = *β*1 (16.11)

|  |
| --- |
| **data**("olympics", package="PoEdata")  olympics.count <- **glm**(medaltot~**log**(pop)+**log**(gdp),  family= "poisson", na.action=na.omit, data=olympics)  **kable**(**tidy**(olympics.count), digits=4, align='c', caption="Poisson model for the 'olympics' problem") |

**library**(AER)

**dispersiontest**(olympics.count)

##

## Overdispersion test

##

*16.9. THE TOBIT, OR CENSORED DATA MODEL*

## data: olympics.count

## z = 5.489, p-value = 2.02e-08

## alternative hypothesis: true dispersion is greater than 1 ## sample estimates:

## dispersion

## 13.5792

Table 16.7 shows the output of a count model to explain the number of medals won by a country based on the country’s population and GDP. The function dispersiontest in package AER tests the validity of the Poisson distribution based on this distribution’s characteristic that its mean is equal to its variance. The null hypothesis of the test is *equidispersion*; rejecting the null questions the validity of the model. Our example fails the overdispersion test.

## 16.9 The Tobit, or Censored Data Model

Censored data include a large number of observations for which the dependent variable takes one, or a limited number of values. An example is the *mroz* data, where about 43 percent of the women observed are not in the labour force, therefore their market hours worked are zero. Figure 16.2 shows the histogram of the variable *wage* in the dataset *mroz*.

**data**("mroz", package="PoEdata") **hist**(mroz$hours, breaks=20, col="grey")

A censored model is based on the idea of a **latent**, or unobserved variable that is not censored, and is explained via a probit model, as shown in Equation 16.12.

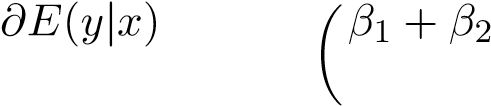
*yi∗* = *β*1 + *β*2*xi* + *ei* (16.12)

The observable variable, *y*, is zero for all *y∗* that are less or equal to zero and is equal to *y∗* when *y∗* is greater than zero. The model for censored data is called **Tobit**, and is described by Equation 16.13.

*P*(*y* = 0) = *P*(*y∗ ≤* 0) = 1 *−* Φ[(*β*1 + *β*2*x*)*/σ*] (16.13)

The marginal effect of a change in *x* on the observed variable *y* is given by Equation

16.14.

*x*

= *β*2Φ (16.14)

*∂x σ*

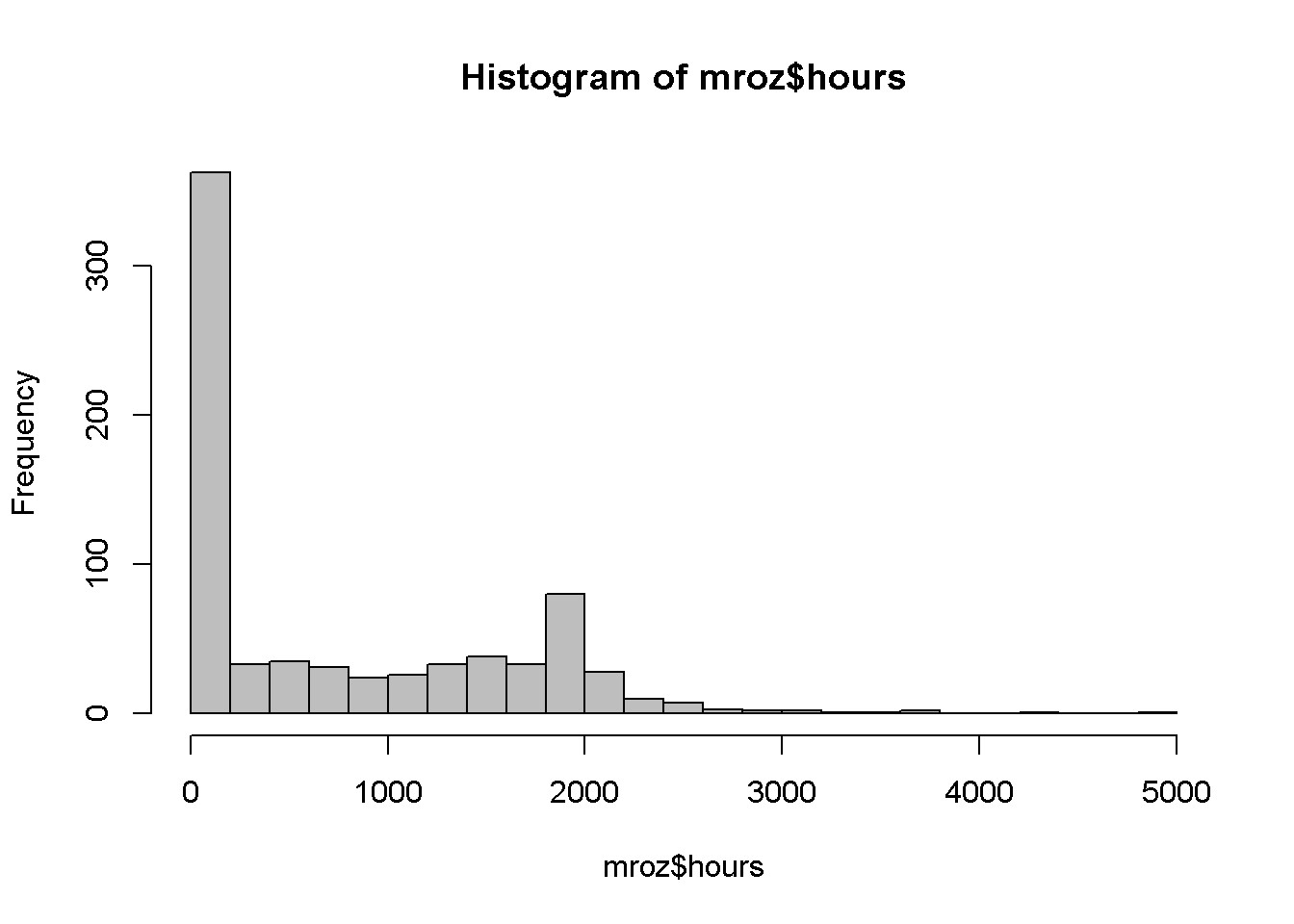


Figure 16.2: Histogram for the variable ’wage’ in the ’mroz’ dataset

**library**(AER)

mroz.tobit <- **tobit**(hours~educ+exper+age+kidsl6, data=mroz)

sMrozTobit <- **summary**(mroz.tobit) sMrozTobit

##

## Call:

## tobit(formula = hours ~ educ + exper + age + kidsl6, data = mroz)

##

## Observations:

## Total Left-censored Uncensored Right-censored

## 753 325 428 0

##

## Coefficients:

## Estimate Std. Error z value Pr(>|z|)

## (Intercept) 1349.8763 386.2991 3.49 0.00048 \*\*\* ## educ 73.2910 20.4746 3.58 0.00034 \*\*\*

## exper 80.5353 6.2878 12.81 < 2e-16 \*\*\* ## age -60.7678 6.8882 -8.82 < 2e-16 \*\*\* ## kidsl6 -918.9181 111.6607 -8.23 < 2e-16 \*\*\*

## Log(scale) 7.0332 0.0371 189.57 < 2e-16 \*\*\*

*16.10. THE HECKIT, OR SAMPLE SELECTION MODEL*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

##

## Scale: 1134

##

## Gaussian distribution

## Number of Newton-Raphson Iterations: 4

## Log-likelihood: -3.83e+03 on 6 Df

## Wald-statistic: 243 on 4 Df, p-value: < 2.2e-16

The following code lines calculate the marginal effect of education on hours for some given values of the regressors.

|  |
| --- |
| xEduc <- 12.29 xExper <- 10.63 xAge <- 42.54 xKids <- 1  bInt <- **coef**(mroz.tobit)[[1]] bEduc <- **coef**(mroz.tobit)[[2]] bExper <- **coef**(mroz.tobit)[[3]] bAge <- **coef**(mroz.tobit)[[4]] bKids <- **coef**(mroz.tobit)[[5]] bSigma <- mroz.tobit$scale  Phactor <- **pnorm**((bInt+bEduc\*xEduc+bExper\*xExper+ bAge\*xAge+bKids\*xKids)/bSigma)  DhoursDeduc <- bEduc\*Phactor |

The calculated marginal effect is 26*.*606. (The function censReg() from package censReg can also be used for estimating Tobit models; this function gives the possibility of calculating marginal effects using the function margEff().)

## 16.10 The Heckit, or Sample Selection Model

The models are useful when the sample selection is not random, but whehter an individual is in the sample depends on individual characteristics. For example, when studying wage determination for married women, some women are not in the labour force, therefore their wages are zero.

The model to use in such situation is **Heckit**, which involves two equations: the **selection equation**, given in Equation 16.15, and the linear **equation of interest**, as in Equation 16.16.

|  |  |
| --- | --- |
| *zi∗* = *γ*1 + *γ*2*wi* + *ui* | (16.15) |
| *yi* = *β*1 + *β*2*xi* + *ei* | (16.16) |

Estimates of the *β*s can be obtained by using least squares on the model in Equation 16.17, where *λi* is calculated using the formula in Equation 16.18.

|  |  |
| --- | --- |
| *yi* = *β*1 + *β*2*xi* + *βλλi* + *νi* | (16.17) |
| *φ*(*γ*1 + *γ*2*wi*)  *λi* = | (16.18) |

Φ(*γ*1 + *γ*2*wi*)

The amount *λ* given by Equation 16.18 is called the **inverse Mills ratio**.

the Heckit procedure involves two steps, estimating both the selection equation and the equation of interest. Function selection() in the sampleSelection package performs both steps; therefore, it needs both equations among its arguments. (The selection equation is, in fact, a *probit* model.)

|  |
| --- |
| **library**(sampleSelection)  wage.heckit <- **selection**(lfp~age+educ+**I**(kids618+kidsl6)+mtr,  **log**(wage)~educ+exper, data=mroz, method="ml")  **summary**(wage.heckit) |

## -------------------------------------------## Tobit 2 model (sample selection model)

## Maximum Likelihood estimation

## Newton-Raphson maximisation, 4 iterations

## Return code 2: successive function values within tolerance limit

## Log-Likelihood: -913.513

## 753 observations (325 censored and 428 observed) ## 10 free parameters (df = 743) ## Probit selection equation:

## Estimate Std. error t value Pr(> t)

## (Intercept) 1.53798 0.61889 2.49 0.013 \* ## age -0.01346 0.00603 -2.23 0.026 \*

## educ 0.06278 0.02180 2.88 0.004 \*\*

## I(kids618 + kidsl6) -0.05108 0.03276 -1.56 0.119 ## mtr -2.20864 0.54620 -4.04 0.000053 \*\*\*

## Outcome equation:

*16.10. THE HECKIT, OR SAMPLE SELECTION MODEL*

## Estimate Std. error t value Pr(> t)

## (Intercept) 0.64622 0.23557 2.74 0.0061 \*\*

## educ 0.06646 0.01657 4.01 0.000061 \*\*\*

## exper 0.01197 0.00408 2.93 0.0034 \*\* ## Error terms:

## Estimate Std. error t value Pr(> t)

## sigma 0.8411 0.0430 19.6 <2e-16 \*\*\*

## rho -0.8277 0.0391 -21.2 <2e-16 \*\*\*

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## --------------------------------------------

250 *CHAPTER 16. QUALITATIVE AND LDV MODELS*

# References

Adkins, Lee. 2014. *Using Gretl for Principles of Econometrics, 4th Edition*. Economics Working Paper Series. 1412. Oklahoma State University, Department of Economics; Legal Studies in Business. [http://EconPapers.repec.org/RePEc:okl: wpaper:1412.](http://EconPapers.repec.org/RePEc:okl:wpaper:1412)

Allaire, JJ, Joe Cheng, Yihui Xie, Jonathan McPherson, Winston Chang, Jeff Allen, Hadley Wickham, Aron Atkins, and Rob Hyndman. 2016. *Rmarkdown: Dynamic Documents for R*. [http://rmarkdown.rstudio.com.](http://rmarkdown.rstudio.com/)

Colonescu, Constantin. 2016. *PoEdata: PoE Data for R*.

Croissant, Yves, and Giovanni Millo. 2015. *Plm: Linear Models for Panel Data*.

[https://CRAN.R-project.org/package=plm.](https://CRAN.R-project.org/package=plm)

Dahl, David B. 2016. *Xtable: Export Tables to Latex or Html*. [https://CRAN. R-project.org/package=xtable.](https://CRAN.R-project.org/package=xtable)

Fox, John, and Sanford Weisberg. 2016. *Car: Companion to Applied Regression*. [https://CRAN.R-project.org/package=car.](https://CRAN.R-project.org/package=car)

Fox, John, Sanford Weisberg, Michael Friendly, and Jangman Hong. 2016. *Effects:*

*Effect Displays for Linear, Generalized Linear, and Other Models*. [https://CRAN. R-project.org/package=effects.](https://CRAN.R-project.org/package=effects)

Ghalanos, Alexios. 2015. *Rugarch: Univariate Garch Models*. [https://CRAN. R-project.org/package=rugarch.](https://CRAN.R-project.org/package=rugarch)

Graves, Spencer. 2014. *FinTS: Companion to Tsay (2005) Analysis of Financial Time Series*. [https://CRAN.R-project.org/package=FinTS.](https://CRAN.R-project.org/package=FinTS)

Grolemund, Garrett, and Hadley Wickham. 2016. *R for Data Science*. Online book. [http://r4ds.had.co.nz/index.html.](http://r4ds.had.co.nz/index.html)

Henningsen, Arne, and Jeff D. Hamann. 2015. *Systemfit: Estimating Systems of Simultaneous Equations*. [https://CRAN.R-project.org/package=systemfit.](https://CRAN.R-project.org/package=systemfit)

Hill, R.C., W.E. Griffiths, and G.C. Lim. 2011. *Principles of Econometrics*. Wiley.

251

252 *CHAPTER 16. QUALITATIVE AND LDV MODELS*

[https://books.google.ie/books?id=Q-fwbwAACAAJ.](https://books.google.ie/books?id=Q-fwbwAACAAJ)

Hlavac, Marek. 2015. *Stargazer: Well-Formatted Regression and Summary Statistics Tables*. [https://CRAN.R-project.org/package=stargazer.](https://CRAN.R-project.org/package=stargazer)

Hothorn, Torsten, Achim Zeileis, Richard W. Farebrother, and Clint Cummins. 2015. *Lmtest: Testing Linear Regression Models*. [https://CRAN.R-project.org/package= lmtest.](https://CRAN.R-project.org/package=lmtest)

Hyndman, Rob. 2016. *Forecast: Forecasting Functions for Time Series and Linear Models*. [https://CRAN.R-project.org/package=forecast.](https://CRAN.R-project.org/package=forecast)

Kleiber, Christian, and Achim Zeileis. 2015. *AER: Applied Econometrics with R*. [https://CRAN.R-project.org/package=AER.](https://CRAN.R-project.org/package=AER)

Komashko, Oleh. 2016. *NlWaldTest: Wald Test of Nonlinear Restrictions and Nonlinear Ci*. [https://CRAN.R-project.org/package=nlWaldTest.](https://CRAN.R-project.org/package=nlWaldTest)

Lander, Jared P. 2013. *R for Everyone: Advanced Analytics and Graphics*. 1st ed. Addison-Wesley Professional.

Lumley, Thomas, and Achim Zeileis. 2015. *Sandwich: Robust Covariance Matrix Estimators*. [https://CRAN.R-project.org/package=sandwich.](https://CRAN.R-project.org/package=sandwich)

Pfaff, Bernhard. 2013. *Vars: VAR Modelling*. [https://CRAN.R-project.org/ package=vars.](https://CRAN.R-project.org/package=vars)

R Development Core Team. 2008. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. [http://www. R-project.org.](http://www.R-project.org/)

Reinhart, Abiel. 2015. *Pdfetch: Fetch Economic and Financial Time Series Data from Public Sources*. [https://CRAN.R-project.org/package=pdfetch.](https://CRAN.R-project.org/package=pdfetch)

Robinson, David. 2016. *Broom: Convert Statistical Analysis Objects into Tidy Data Frames*. [https://CRAN.R-project.org/package=broom.](https://CRAN.R-project.org/package=broom)

RStudio Team. 2015. *RStudio: Integrated Development Environment for R*. Boston, MA: RStudio, Inc. [http://www.rstudio.com/.](http://www.rstudio.com/)

Spada, Stefano, Matteo Quartagno, and Marco Tamburini. 2012. *Orcutt: Estimate Procedure in Case of First Order Autocorrelation*. [https://CRAN.R-project.org/ package=orcutt.](https://CRAN.R-project.org/package=orcutt)

Trapletti, Adrian, and Kurt Hornik. 2016. *Tseries: Time Series Analysis and Computational Finance*. [https://CRAN.R-project.org/package=tseries.](https://CRAN.R-project.org/package=tseries)

Wickham, Hadley, and Winston Chang. 2016. *Devtools: Tools to Make Developing*

*16.10. THE HECKIT, OR SAMPLE SELECTION MODEL* 253

*R Packages Easier*. [https://CRAN.R-project.org/package=devtools.](https://CRAN.R-project.org/package=devtools)

Xie, Yihui. 2014. *Printr: Automatically Print R Objects According to Knitr Output Format*. [http://yihui.name/printr.](http://yihui.name/printr)

———. 2016a. *Bookdown: Authoring Books with R Markdown*. [https://CRAN.](https://CRAN.R-project.org/package=bookdown)

[R-project.org/package=bookdown.](https://CRAN.R-project.org/package=bookdown)

———. 2016b. *Knitr: A General-Purpose Package for Dynamic Report Generation in R*. [https://CRAN.R-project.org/package=knitr.](https://CRAN.R-project.org/package=knitr)

Zeileis, Achim. 2016. *Dynlm: Dynamic Linear Regression*. [https://CRAN.R-project. org/package=dynlm.](https://CRAN.R-project.org/package=dynlm)