

**Faculty of Business and Law**

**School of Accounting, Finance and Economics**

**BECS2002 / Econometrics and Data Analytics**

**Lab Handbook**

**Week 15**

**Module coordinator:** Dr Camilo Calderon

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

**This version: 21/02/2025**

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

Prerequisite 6

Lecture 1 - The Simple Linear Regression Model 7

Lecture 2 - Prediction with the Linear Regression Model 15

Video 1 - Using Indicator Variables in a Regression & Monte Carlo 23

Video 2 -Interval Estimation 28

Lab 1 – Hypothesis Test, P-Value and Testing Linear Combination of Parameters 35

Lab 2 -The *p*-Value 40

Lab 3 - Prediction, R-squared, and Modelling 45

Lab 4 Linear-Log Models 52

Lab 5- Polynomial Models 59

Lab 6 – Hands-on on your assignment 68

Prerequisite

1. Review the assignment brief for the report assessment and download an appropriate dataset.

Students will adapt the R scripts of the next three weeks of labs to the dataset and topic of the report.

Lecture 1 - The Simple Linear Regression Model

**Note:** students must adapt the following R scrips to their assignment 2 (report) using an appropriate dataset.

**Summary:** in simple linear regression, we use one independent variable to predict a dependent variable. The relationship is represented by a straight line with an intercept and a slope.

**Introduction:** In this lecture, we will explore the fundamentals of the Simple Linear Regression Model, a statistical method used to understand the relationship between two variables. This model is foundational for understanding more complex statistical analyses in your future studies.

**Key terms:** A dependent variable, represented as 'y', is what we aim to predict, while an independent variable, denoted as 'x', is the variable(s) we use to explain y. Imagine predicting a student's final grade (y) based on their hours of study (x). Here, the final grade depends on, or is predicted by, the hours of study.

## The General Model

The model assumes a linear relationship between the conditional expectation of a dependent variable, *y*, and an independent variable, *x*. Independent variable sometimes are called ‘response’ or ‘response variable’, and the independent variables ‘regressors.’ The assumed relationship has the form:

*yi* = *β*1 + *β*2*xi* + *ei, (1)*

where

* *y* is the *dependent variable*
* *x* is the *independent variable*
* *e* is an *error term*
* *σ*2 is the variance of the error term
* *β*1 is the *intercept* parameter or coefficient
* *β*2 is the *slope* parameter or coefficient
* *i* stands fot the *i* -th observation in the dataset, *i* = 1*,*2*,...,N*
* *N* is the number of observations in the dataset

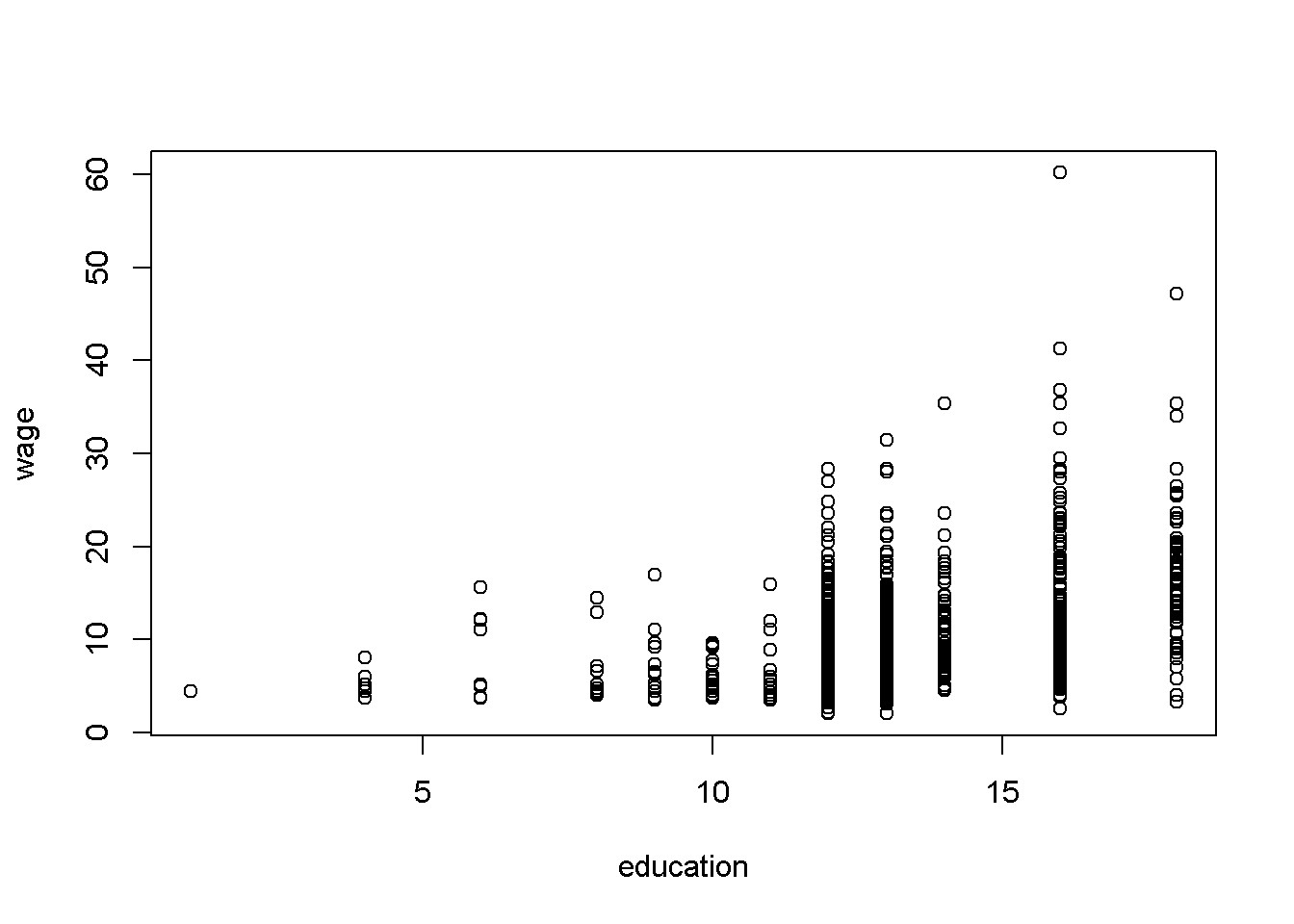


Figure: Example of several observations for any given *x*

The *predicted*, or estimated value of *y* given *x* is given by the following equation; in general, the *hat* symbol indicates an estimated or a predicted value.

*y*ˆ = *b*1 + *b*2*x (2)*

The model assumes that the values of *x* are previously chosen (therefore, they are non-random), that the variance of the error term, *σ*2, is the same for all values of *x*, and that there is no connection between one observation and another (no correlation between the error terms of two observations). In addition, it is assumed that the expected value of the error term for any value of *x* is zero.

The subscript *i* in Equation 1 indicates that the relationship applies to each of the *N* observations. Thus, there must be specific values of *y*, *x*, and *e* for each observation. However, since *x* is not random, there are, typically, several observations sharing the same *x*, as the scatter diagram in Figure above shows.

|  |
| --- |
| **rm**(list=ls()) *# Caution: this clears the Environment*  **install.packages(“devtools”)**  **install.packages(“remotes”)**  **remotes::install\_github(“ccolonescu/PoEdata”)**  **library**(PoEdata) *# Makes datasets ready to use*  **data**("cps\_small")  **plot**(cps\_small$educ, cps\_small$wage, xlab="education", ylab="wage") |

## Example: Food Expenditure versus Income

The data for this example is stored in the R package PoEdata (To check if the package PoEdata is installed, look in the Packages list.)

**library**(PoEdata)

**data**(food)

**head**(food)

|  |  |
| --- | --- |
| food\_exp | income |
| 115.22 | 3.69 |
| 135.98 | 4.39 |
| 119.34 | 4.75 |
| 114.96 | 6.03 |
| 187.05 | 12.47 |
| 243.92 | 12.98 |

It is always a good idea to visually inspect the data in a **scatter diagram**, which can be created using the function plot(). Figure 2.2 is a scatter diagram of food expenditure on income, suggesting that there is a positive relationship between income and food expenditure.

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

**plot**(food$income, food$food\_exp, ylim=**c**(0, **max**(food$food\_exp)), xlim=**c**(0, **max**(food$income)), xlab="weekly income in $100", ylab="weekly food expenditure in $", type = "p")

## Estimating a Linear Regression

The R function for estimating a linear regression model is lm(y~x, data) which, used just by itself does not show any output; It is useful to give the model a name, such as mod1, then show the results using summary(mod1). If you are interested in only some of the results of the regression, such as the estimated coefficients, you can retrieve them using specific functions, such as the function coef(). For the food expenditure data, the regression model will be

*food*\_*exp* = *β*1 + *β*2*income* + *e* (3)

where the subscript *i* has been omitted for simplicity.



Figure: A scatter diagram for the food expenditure model

**library**(PoEdata)

mod1 <- **lm**(food\_exp ~ income, data = food)

b1 <- **coef**(mod1)[[1]]

b2 <- **coef**(mod1)[[2]]

smod1 <- **summary**(mod1)

smod1

##

## Call:

## lm(formula = food\_exp ~ income, data = food)

##

## Residuals:

## Min 1Q Median 3Q Max

## -223.03 -50.82 -6.32 67.88 212.04

##

## Coefficients:

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

## (Intercept) 83.42 43.41 1.92 0.062 .

## income 10.21 2.09 4.88 0.000019 \*\*\*

## ---

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

##

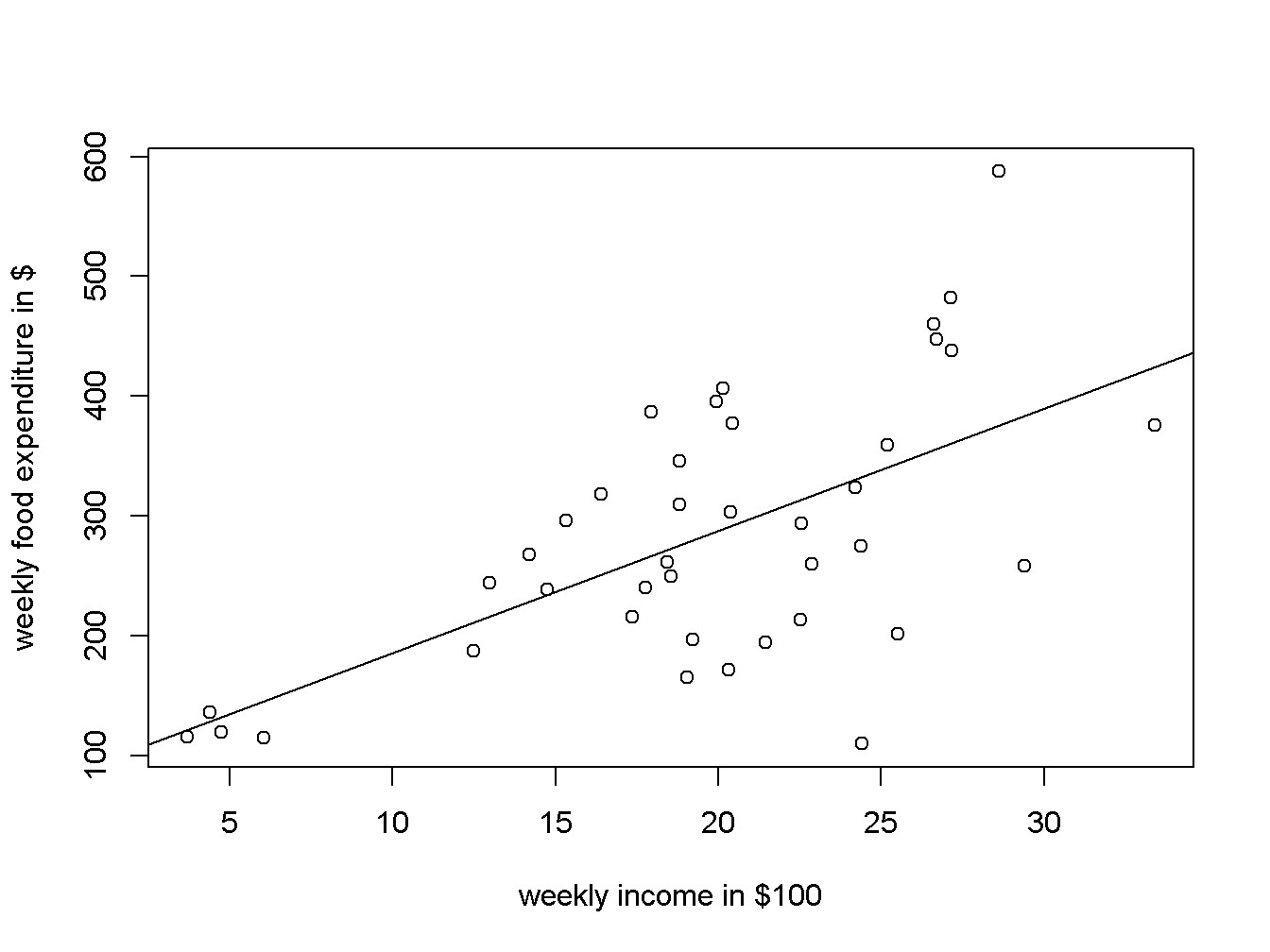


Figure: Scatter diagram and regression line for the food expenditure model

## Residual standard error: 89.5 on 38 degrees of freedom

## Multiple R-squared: 0.385, Adjusted R-squared: 0.369

## F-statistic: 23.8 on 1 and 38 DF, p-value: 0.0000195

The function coef() returns a list containing the estimated coefficients. You can access individual coefficients. The estimated value of *β*1 is b1 <- coef(mod1)[[1]], which is equal to 83.416002, and the estimated value of *β*2 is b2 <- coef(mod1)[[2]], which is equal to 10.209643.

The intercept parameter, *β*1, usually has little importance in econometric models. The estimated value of *β*2 suggests that the food expenditure for an average family increases by 10.209643 when the family income increases by 1 unit, which in this case is $100. See the coefficients in the estimation results table.

The *R* function abline() adds the regression line to the previously plotted scatter diagram, as the Figure above shows.

|  |
| --- |
| **plot**(food$income, food$food\_exp, ylim=**c**(0, **max**(food$food\_exp)), xlim=**c**(0, **max**(food$income)), xlab="weekly income in $100", ylab="weekly food expenditure in $", type = "p")  **abline**(b1,b2) |

How can one retrieve various regression results? These results exist in two R *objects* produced by the lm() function: the regression object, such as mod1 in the above code sequence, and the regression summary, which I denoted by smod1. The next code shows how to list the names of all results in each object.

|  |  |  |
| --- | --- | --- |
| **names**(mod1) |  |  |
| ## [1] "coefficients" "residuals" | "effects" | "rank" |
| ## [5] "fitted.values" "assign" | "qr" | "df.residual" |
| ## [9] "xlevels" "call" | "terms" | "model" |
| **names**(smod1) |  |  |

## [1] "call" "terms" "residuals" "coefficients"

## [5] "aliased" "sigma" "df" "r.squared"

## [9] "adj.r.squared" "fstatistic" "cov.unscaled"

To retrieve a particular result you just refer to it with the name of the object, followed by the $ sign and the name of the result you wish to retrieve. For instance, if we want the vector of coefficients from mod1, we refer to it as **mod1$coefficients** and smod1$coefficients:

mod1$coefficients

## (Intercept) income ## 83.4160 10.2096

smod1$coefficients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t value | Pr(>|t|) |
| (Intercept) | 83.4160 | 43.41016 | 1.92158 | 0.062182 |
| income | 10.2096 | 2.09326 | 4.87738 | 0.000019 |

As we have seen before, however, some of these results can be retrieved using specific functions, such as coef(mod1), resid(mod1), fitted(mod1), and vcov(mod1).

## MCQs

1. What does a simple linear regression model assume about the relationship between the dependent variable *y* and the independent variable *x*?

A. A quadratic relationship.

B. A logarithmic relationship.

C. A linear relationship.

D. An exponential relationship.

1. In the equation , what does represent?

A. The slope of the regression line.

B. The error term.

C. The intercept of the regression line.

D. The dependent variable.

1. What is indicated by the slope parameter in a simple linear regression model?

A. The variance of the error term.

B. The change in *y* for a one-unit change in *x*.

C. The intercept of the regression line.

D. The number of observations in the dataset.

1. In the context of simple linear regression, what does the term "residuals" refer to?

A. The predicted values of *y*.

B. The independent variable values.

C. The differences between the observed and predicted values of *y*.

D. The coefficients of the regression model.

1. When using the R function lm() for linear regression, what is the purpose of the summary() function?

A. To list the names of all results in the regression object.

B. To change the coefficients of the regression model.

C. To plot the regression line on a scatter plot.

D. To display a detailed summary of the regression results, including coefficients, residuals, and R-squared values.

1. What is the primary purpose of estimating a linear regression model using random subsamples in R?

A. To change the coefficients of the original model.

B. To assess the variability of the regression coefficients.

C. To increase the number of observations in the dataset.

D. To create a non-linear relationship between variables.

1. In a simple linear regression model, what does an "Adjusted R-squared" value indicate?

A. The total number of observations in the dataset.

B. The precision of the intercept parameter.

C. The proportion of variance in the dependent variable explained by the model, adjusted for the number of predictors.

D. The standard error of the regression coefficients.

Lecture 2 - Prediction with the Linear Regression Model

The estimated regression parameters, *b*1 and *b*2 allow us to *predict* the expected food expenditure for any given income. All we need to do is to plug the estimated parameter values and the given income into an equation like Equation (2). For example, the expected value of *food*\_*exp* for an income of $2000 is calculated in Equation (4) below. (Remember to divide the income by 100, since the data for the variable *income* is in hundreds of dollars.)

 (4)

R does this calculations for us with its function called predict(). Extending the example to more than one income, say *income* = $2000, $2500, and $2700. The function predict() in

R requires that the new values of the independent variables be organized under a particular form, called a **data frame**. In R, a set of numbers is held together using the structure c(). The following sequence shows this example.

|  |
| --- |
| *# library(PoEdata) (load the data package if you have not done so yet)*  *# This line of code creates a linear model named mod1. The lm() function is used for linear regression. food\_exp~income indicates that we are predicting food\_exp (food expenditure) based on income.* *data=food specifies that the data for this model comes from the food dataset.*  mod1 <- **lm**(food\_exp~income, data=food)  *# Here, we're creating a new data frame newx. The c(20, 25, 27) part is a vector representing new income values (in hundreds of dollars) for which we want to predict food expenditure. data.frame() converts this vector into a data frame, which is the required format for the predict() function.*  newx <- **data.frame**(income = **c**(20, 25, 27))  *# This line uses the predict() function to estimate food expenditure based on our model mod1 and the new income data in newx.*  yhat <- **predict**(mod1, newx)  **names**(yhat) <- **c**("income=$2000", "$2500", "$2700")  yhat *# prints the result* |

## income=$2000 $2500 $2700

## 287.609 338.657 359.076

## Repeated Samples to Assess Regression Coefficients

The regression coefficients *b*1 and *b*2 are random variables because they depend on the sample. Let us construct a number of random subsamples from the *food* data and re-calculate *b*1 and *b*2. A random subsample can be constructed using the function sample(), as the following example illustrates only for *b*2.

|  |
| --- |
| N <- **nrow**(food) *# returns the number of observations in the dataset*  C <- 50 *# desired number of subsamples*  S <- 38 *# desired sample size*  sumb2 <- 0  for (i in 1:C){ *# a loop over the number of subsamples*  **set.seed**(3\*i) *# a different seed for each subsample*  subsample <- food[**sample**(1:N, size=S, replace=TRUE), ]  mod2 <- **lm**(food\_exp~income, data=subsample)  *#sum b2 for all subsamples:*  sumb2 <- sumb2 + **coef**(mod2)[[2]]  }  **print**(sumb2/C, digits = 3) |

## [1] 9.88

The result, *b*2 = 9.88, is the average of 50 estimates of *b*2.

## Estimated Variances and Covariance of Regression Coefficients

Many applications require estimates of the variances and covariances of the regression coefficients. R stores them in the a matrix vcov():

(varb1 <- **vcov**(mod1)[1, 1])

## [1] 1884.44

(varb2 <- **vcov**(mod1)[2, 2])

## [1] 4.38175

(covb1b2 <- **vcov**(mod1)[1,2])

## [1] -85.9032

## Non-Linear Relationships

Sometimes the scatter plot diagram or some theoretical considerations suggest a non-linear relationship. The most popular non-linear relationships involve logarithms of the dependent or independent variables and polynomial functions.

The *quadratic* model requires the square of the independent variable.

*yi* = *β*1 + *β*2*x*2*i* + *ei* (5)

In R, independent variables involving mathematical operators can be included in a regression equation with the function I(). The following example uses the dataset br from the package PoEdata, which includes the sale prices and the attributes of 1080 houses in Baton Rouge, LA. price is the sale price in dollars, and sqft is the surface area in square feet.

|  |
| --- |
| **library**(PoEdata)  **data**(br)  mod3 <- **lm**(price~**I**(sqft^2), data=br)  b1 <- **coef**(mod3)[[1]]  b2 <- **coef**(mod3)[[2]]  sqftx=**c**(2000, 4000, 6000) *#given values for sqft*  pricex=b1+b2\*sqftx^2 *#prices corresponding to given sqft*  DpriceDsqft <- 2\*b2\*sqftx *# marginal effect of sqft on price* |

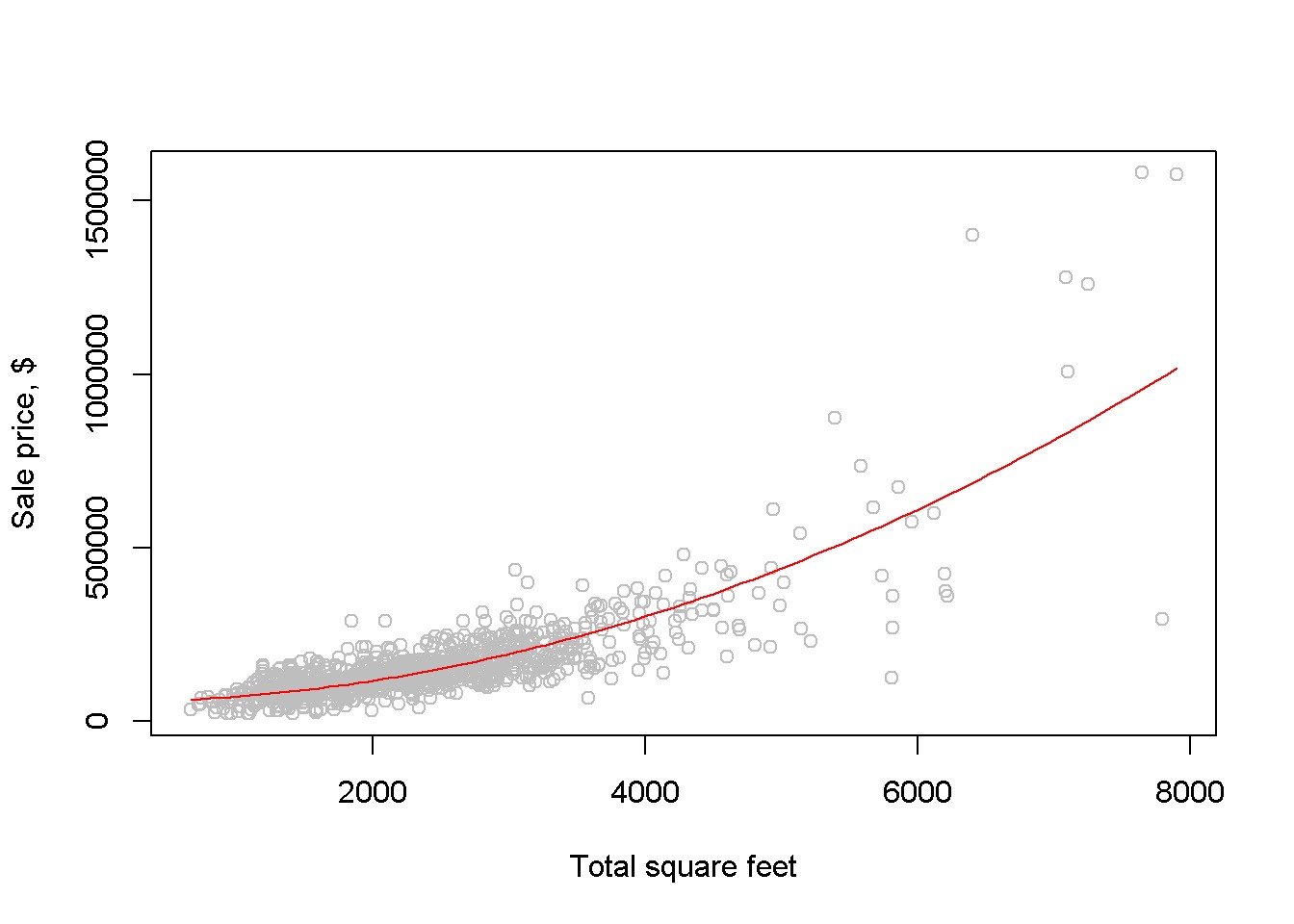


Figure 2.4: Fitting a quadratic model to the ‘br‘ dataset

elasticity=DpriceDsqft\*sqftx/pricex

b1; b2; DpriceDsqft; elasticity *#prints results*

## [1] 55776.6

## [1] 0.0154213

## [1] 61.6852 123.3704 185.0556

## [1] 1.05030 1.63125 1.81741

**Marginal Effect:** it measures the change in the dependent variable (e.g., price) for a unit change in an independent variable (e.g., sqft)

**Elasticity:** It measures the percentage change in the dependent variable (e.g., price) for a one percent change in an independent variable (e.g., sqft). For instance, if the elasticity of price with respect to sqft is 0.5, a 1% increase in sqft leads to a 0.5% increase in price. if the elasticity is less than 1, it suggests that changes in sqft have a proportionally smaller effect on price, which might imply that other factors are also important in determining the price. unlike linear models, in non-linear models, elasticity can change at different levels of the independent variable.

Draw a scatter diagram and see how the quadratic function fits the data. The next chunk of code provides two alternatives for constructing such a graph. The first simply draws the quadratic function on the scatter diagram, using the R function curve(); the second uses the function lines, which requires ordering the dataset in increasing values of sqft before the regression model is evaluated, such that the resulting fitted values will also come out in the same order.

mod31 <- **lm**(price~**I**(sqft^2), data=br)

**plot**(br$sqft, br$price, xlab="Total square feet", ylab="Sale price, $", col="grey")

*#add the quadratic curve to the scatter plot:*

**curve**(b1+b2\*x^2, col="red", add=TRUE)

An alternative way to draw the fitted curve:

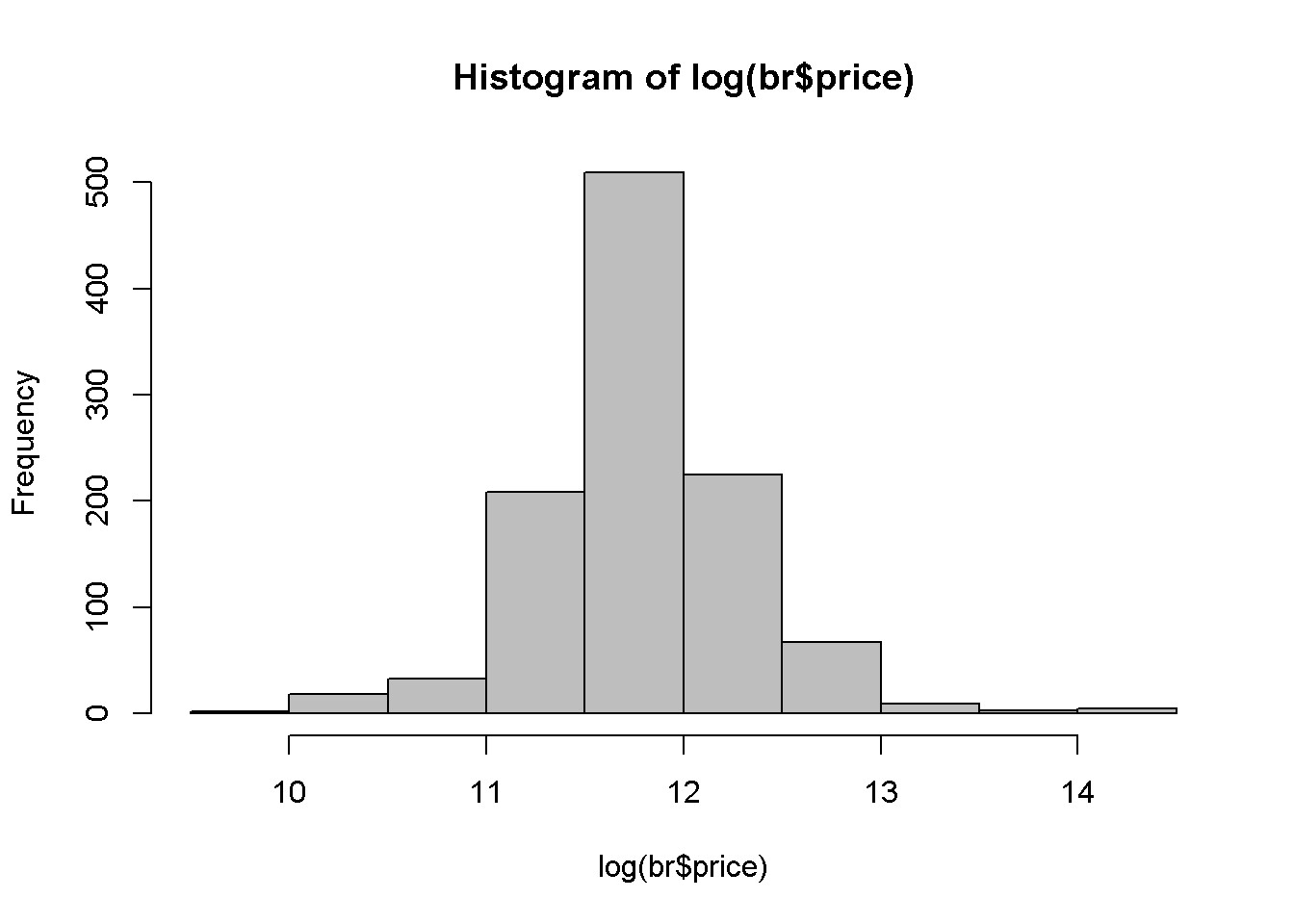
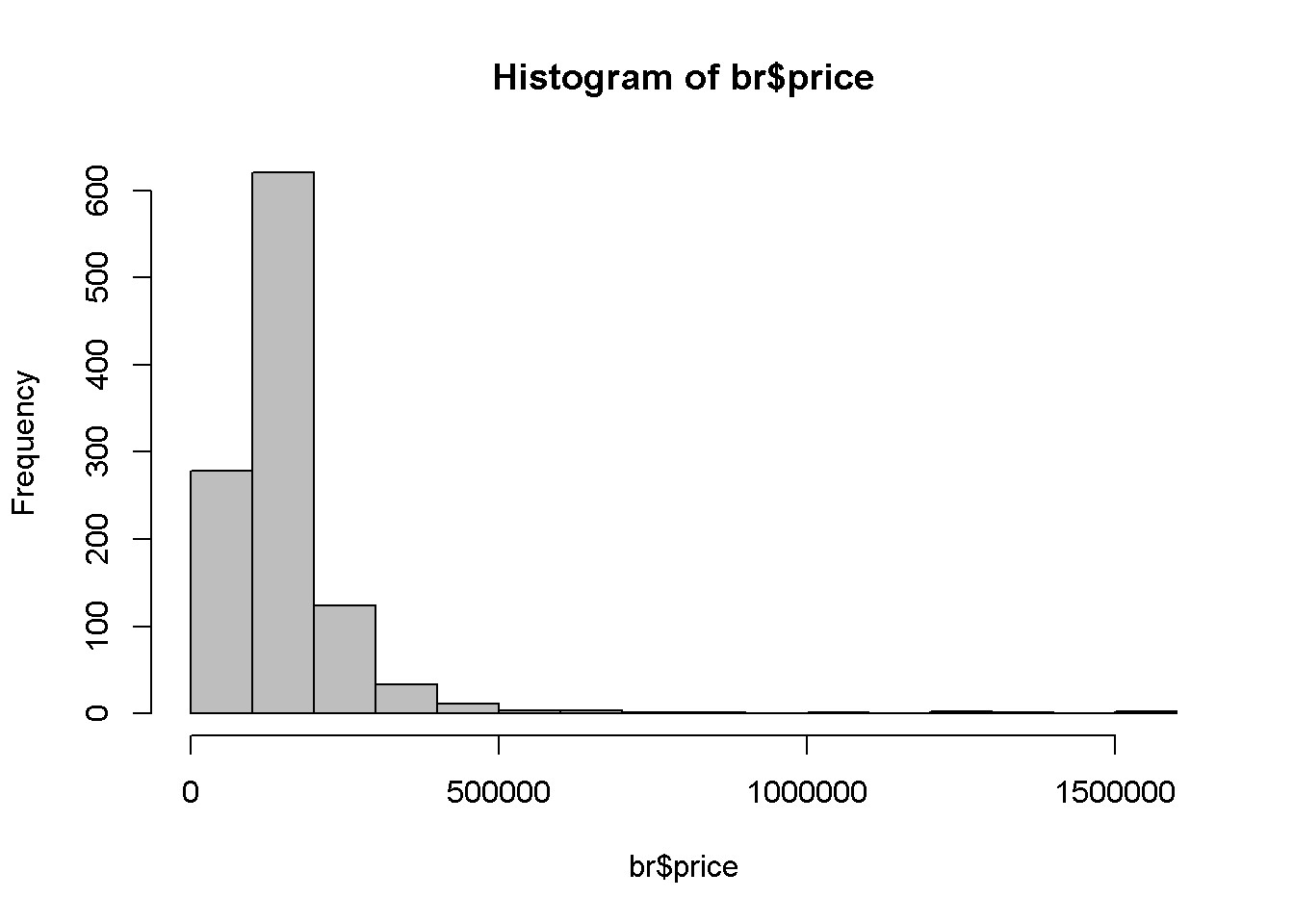


Figure: A comparison between the histograms of ‘price‘ and ‘log(price)‘

|  |
| --- |
| ordat <- br[**order**(br$sqft), ] *#sorts the dataset after `sqft`*  mod31 <- **lm**(price~**I**(sqft^2), data=ordat)  **plot**(br$sqft, br$price, main="Dataset ordered after 'sqft' ", xlab="Total square feet", ylab="Sale price, $", col="grey")  **lines**(**fitted**(mod31)~ordat$sqft, col="red") |

The *log-linear* model regresses the log of the dependent variable on a linear expression of the independent variable (unless otherwise specified, the log notation stands for natural logarithm, following a usual convention in economics):

*log*(*yi*) = *β*1 + *β*2*xi* + *ei* (6)

One of the reasons to use the *log* of an independent variable is to make its distribution closer to the normal distribution. Let us draw the histograms of price and log(price) to compare them (see last Figure above). It can be noticed that that the log is closer to the normal distribution.

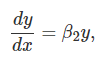
**hist**(br$price, col='grey')

**hist**(**log**(br$price), col='grey')

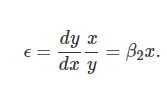
We are interested, as before, in the estimates of the coefficients and their interpretation, in the fitted values of price, and in the marginal effect of an increase in sqft on price.

|  |
| --- |
| **library**(PoEdata)  **data**("br")  mod4 <- **lm**(**log**(price)~sqft, data=br) |

The coefficients are *b*1 = 10*.*84 and *b*2 = 0*.*00041, showing that an increase in the surface area (sqft) of an apartment by one unit (1 sqft) increases the price of the apartment by 0*.*041 percent. Thus, for a house price of $100,000, an increase of 100 sqft will increase the price by approximately 100 *∗* 0*.*041 percent, which is equal to $4112.7. In general, the marginal effect of an increase in *x* on *y* in Equation 6 is



and the elasticity is



The next lines of code show how to draw the fitted values curve of the loglinear model and how to calculate the marginal effect and the elasticity for the median price in the dataset. The fitted values are here calculated using the formula

(9)

ordat <- br[**order**(br$sqft), ] *#order the dataset*

mod4 <- **lm**(**log**(price)~sqft, data=ordat)

**plot**(br$sqft, br$price, col="grey")

**lines**(**exp**(**fitted**(mod4))~ordat$sqft, col="blue", main="Log-linear Model")

pricex<- **median**(br$price)

sqftx <- (**log**(pricex)-**coef**(mod4)[[1]])/**coef**(mod4)[[2]]

(DyDx <- pricex\***coef**(mod4)[[2]])

## [1] 53.465

(elasticity <- sqftx\***coef**(mod4)[[2]])

## [1] 0.936693

R allows us to calculate the same quantities for several *(sqft, price)* pairs at a time, as shown in the following sequence:

|  |
| --- |
| b1 <- **coef**(mod4)[[1]]  b2 <- **coef**(mod4)[[2]]  *#pick a few values for sqft:*  sqftx <- **c**(2000, 3000, 4000)  *#estimate prices for those and add one more:*  pricex <- **c**(100000, **exp**(b1+b2\*sqftx))  *#re-calculate sqft for all prices:*  sqftx <- (**log**(pricex)-b1)/b2  *#calculate and print elasticities:*  (elasticities <- b2\*sqftx)  ## [1] 0.674329 0.822538 1.233807 1.645075 |

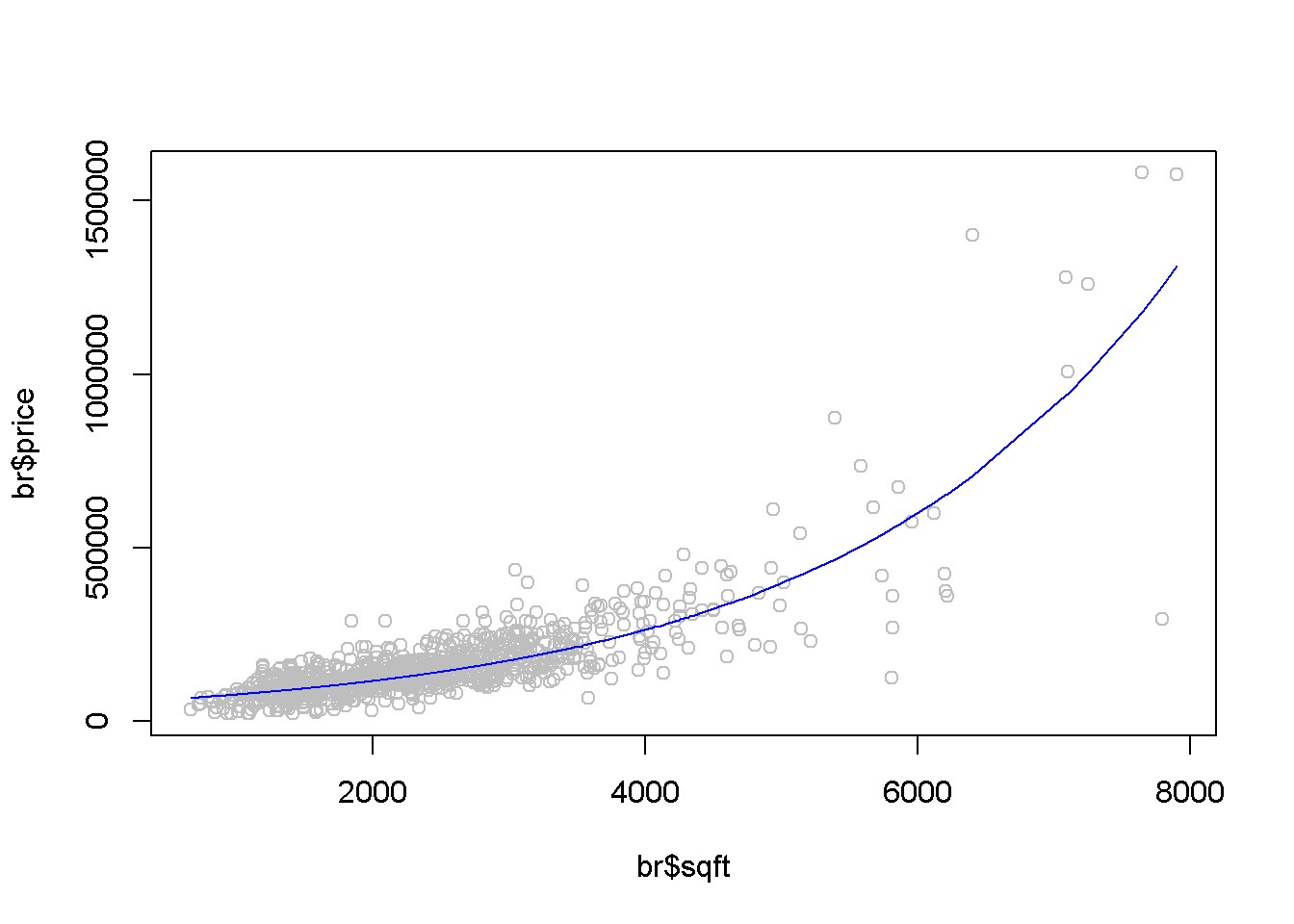


Figure: The fitted value curve in the log-linear model

### MCQs

1 What is the primary purpose of using the estimated regression parameters and in a linear regression model?

A. To calculate the mean of the dependent variable.

B. To predict the expected value of a dependent variable for a given independent variable.

C. To determine the correlation between independent and dependent variables.

D. To assess the distribution of the independent variable.

2 In the context of R programming, what is the function of the lm() function when used in linear regression analysis?

A. To plot the regression line on a scatter plot.

B. To create a new data frame.

C. To estimate a linear model.

D. To calculate the mean of a dataset.

3. What does the predict() function in R primarily do in the context of linear regression?

A. It calculates the variance of the regression coefficients.

B. It estimates the dependent variable values based on the regression model and new data.

C. It generates random samples for assessing regression coefficients.

D. It plots a scatter diagram for visual inspection.

4. Why are regression coefficients and considered random variables?

A. Because they vary depending on the chosen statistical test.

B. Because they are calculated using random number generators.

C. Because they depend on the specific sample used in the regression.

D. Because they change with each iteration of the regression model.

5. What is the purpose of using random subsamples in the assessment of regression coefficients?

A. To visualize the data distribution.

B. To predict values for a new dataset.

C. To evaluate the stability and variability of the coefficients.

D. To calculate the mean and median of the dataset.

6. In the context of regression analysis, what is the vcov() matrix in R used for?

A. For storing predicted values.

B. For estimating variances and covariances of regression coefficients.

C. For plotting regression lines.

D. For generating random subsamples.

7. What is the significance of using the **data.frame()** function in conjunction with the **predict()** function in R?

A. To format the independent variable values appropriately for prediction.

B. To calculate the correlation coefficient between variables.

C. To store the regression coefficients.

D. To create a scatter plot of the data​

bcbccba

Video 1 - Using Indicator Variables in a Regression & Monte Carlo

An indicator or binary variable marks the presence or the absence of some attribute, such as gender or race if the observational unit is an individual or location if the observational unit is a house. In the dataset *utown*, the variable *utown* is 1 if a house is close to the university and 0 otherwise. Here is a simple linear regression model that involves the variable utown:

*pricei* = *β*1 + *β*2*utowni* (10)

The coefficient of such a variable in a simple linear model is equal to the difference between the average prices of the two categories; the intercept coefficient of the model in Equation 10 is equal to the average price of the houses that are not close to university. calculate the average prices for each category, which are denoted in the following sequence of code *price0bar* and *price1bar*:

*# Load the 'utown' dataset which contains housing data, including proximity to a university*

**data**(utown)

*# Calculate the average price of houses not close to the university*

price0bar <- **mean**(utown$price[**which**(utown$utown==0)])

*# Calculate the average price of houses close to the university*

price1bar <- **mean**(utown$price[**which**(utown$utown==1)])

The = 277*.*24 for close to university, and = 215*.*73 for those not close. I now show that the same results yield the coefficients of the regression model in Equation 10:

*# Perform linear regression with 'utown' as an indicator variable*

mod5 <- **lm**(price~utown, data=utown)

*# Extract coefficients from the regression model*

b1 <- **coef**(mod5)[[1]] *#intercept*

b2 <- **coef**(mod5)[[2]] *# Slope for the indicator variable*

The results are: = *b*1 = 215*.*73 for non-university houses, and = *b*1 +*b*2 = 277*.*24 for university houses.

## Monte Carlo Simulation

A Monte Carlo simulation generates random values for the dependent variable when the regression coefficients and the distribution of the random term are given. The following example seeks to determine the distribution of the independent variable in the food expenditure model in Equation 3.

|  |
| --- |
| N <- 40  x1 <- 10  x2 <- 20  b1 <- 100  b2 <- 10  mu <- 0  sig2e <- 2500  sde <- **sqrt**(sig2e)  yhat1 <- b1+b2\*x1  yhat2 <- b1+b2\*x2  **curve**(**dnorm**(x, mean=yhat1, sd=sde), 0, 500, col="blue")  **curve**(**dnorm**(x, yhat2, sde), 0,500, add=TRUE, col="red")  **abline**(v=yhat1, col="blue", lty=2)  **abline**(v=yhat2, col="red", lty=2)  **legend**("topright", legend=**c**("f(y|x=10)","f(y|x=20)"), lty=1,  col=**c**("blue", "red")) |

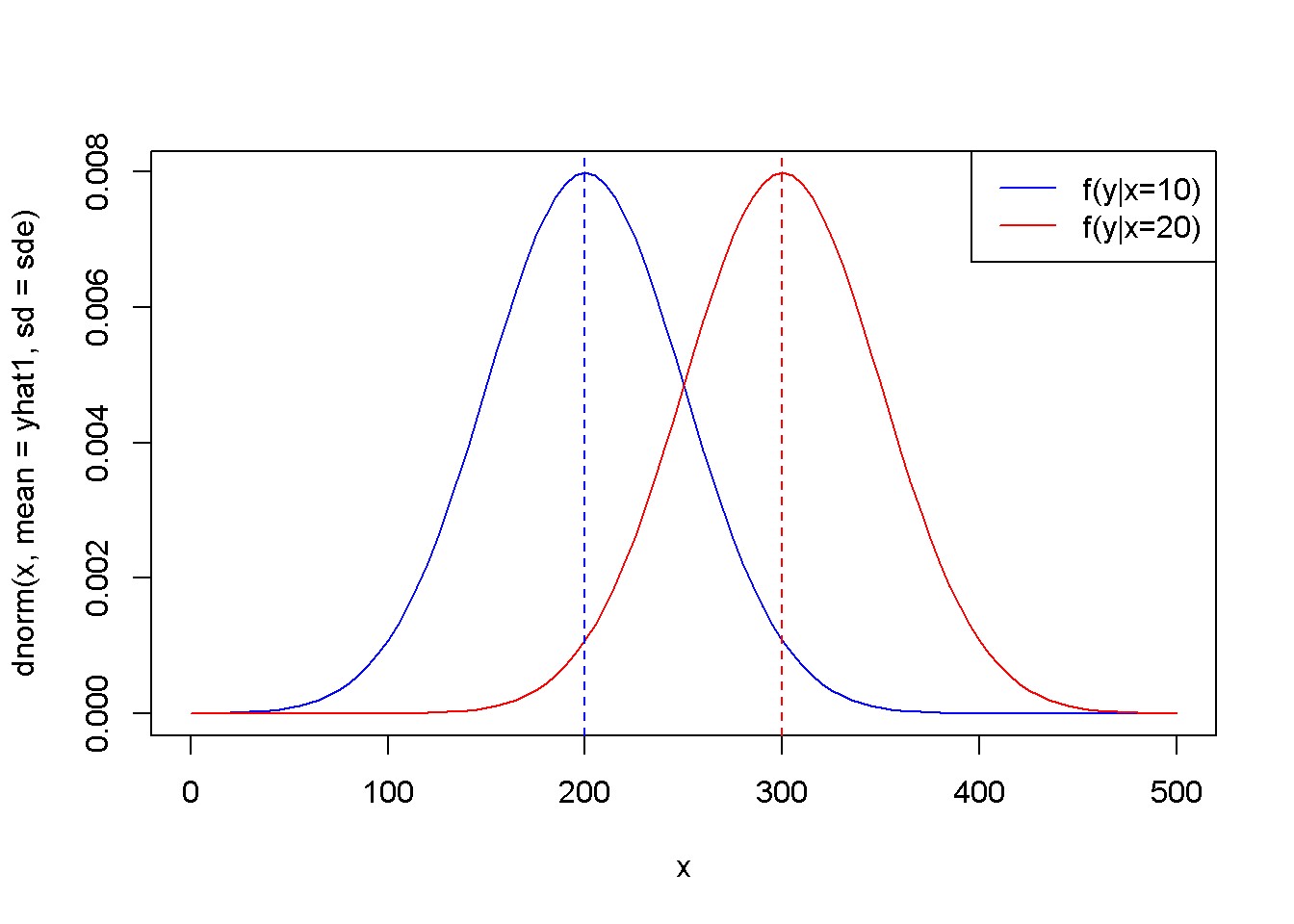
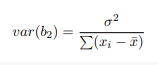
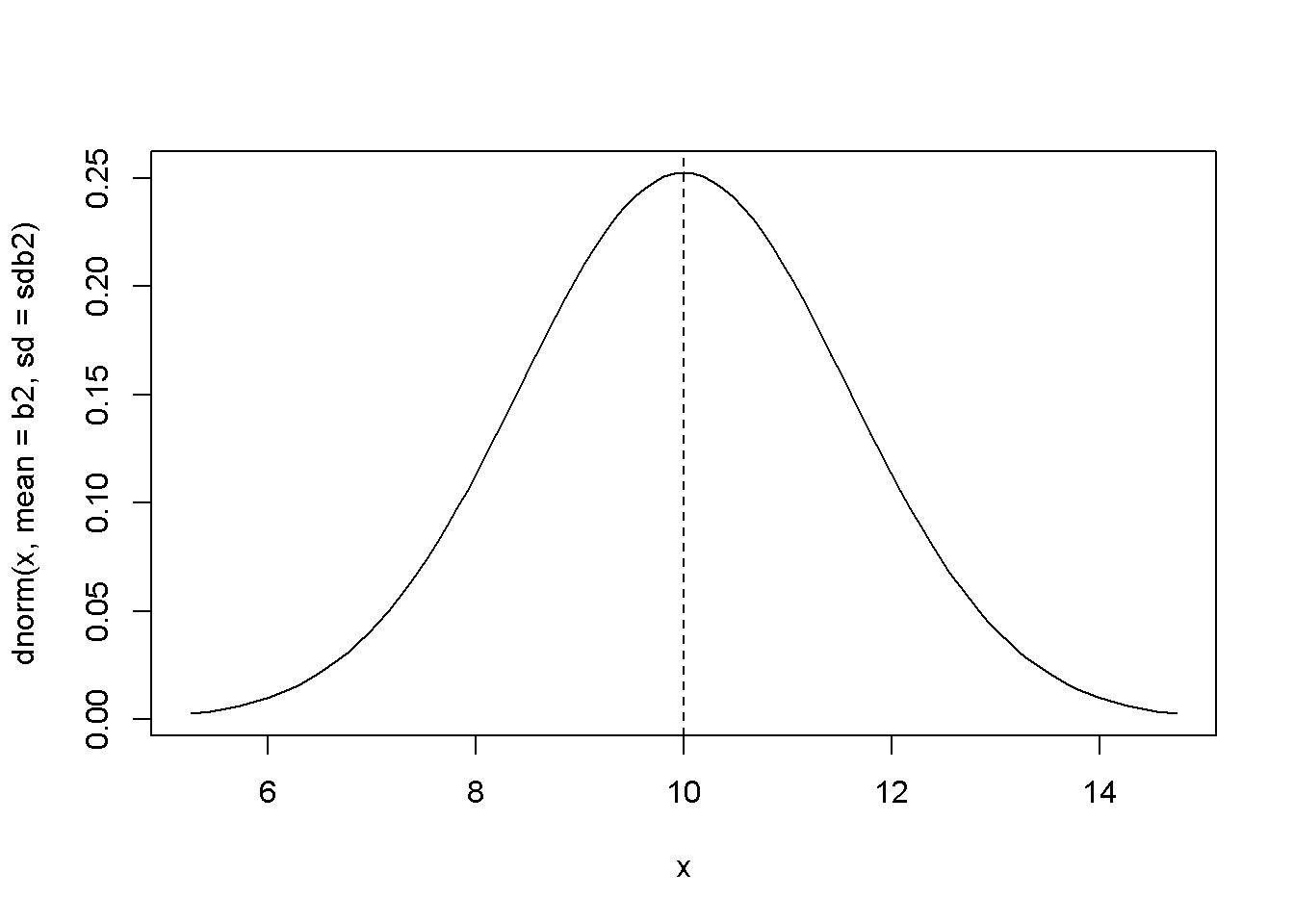


Figure: The theoretical (true) probability distributions of food expenditure, given two levels of income

Next, we calculate the variance of *b*2 and plot the corresponding density function.

 (11)

|  |
| --- |
| x <- **c**(**rep**(x1, N/2), **rep**(x2,N/2))  xbar <- **mean**(x)  sumx2 <- **sum**((x-xbar)^2)  varb2 <- sig2e/sumx2  sdb2 <- **sqrt**(varb2)  leftlim <- b2-3\*sdb2  rightlim <- b2+3\*sdb2  **curve**(**dnorm**(x, mean=b2, sd=sdb2), leftlim, rightlim)  **abline**(v=b2, lty=2) |



Now, with the same values of *b*1, *b*2, and error standard deviation, we can generate a set of values for *y*, regress *y* on *x*, and calculate an estimated value for the coefficient *b*2 and its standard error.

**set.seed**(12345)

y <- b1+b2\*x+**rnorm**(N, mean=0, sd=sde)

mod6 <- **lm**(y~x)

b1hat <- **coef**(mod6)[[1]]

b2hat <- **coef**(mod6)[[2]]

mod6summary <- **summary**(mod6) *#the summary contains the standard errors* seb2hat <- **coef**(mod6summary)[2,2]

The results are *b*2 = 11.64 and *se*(*b*2) = 1.64. The strength of a Monte Carlo simulation is the possibility of repeating the estimation of the regression parameters for a large number of automatically generated samples. Thus, we can obtain a large number of values for a parameter, say *b*2, and then determine its sampling characteristics. For instance, if the mean of these values is close to the initially assumed value *b*2 = 10, we conclude that our estimator (the method of estimating the parameter) is unbiased.

We are going to use this time the values of *x* in the *food* dataset, and generate *y* using the linear model with *b*1 = 100 and *b*2 = 10.

|  |
| --- |
| **data**("food")  N <- 40  sde <- 50  x <- food$income  nrsim <- 1000  b1 <- 100  b2 <- 10  vb2 <- **numeric**(nrsim) *#stores the estimates of b2*  for (i in 1:nrsim){ |
| **set.seed**(12345+10\*i)  y <- b1+b2\*x+**rnorm**(N, mean=0, sd=sde)  mod7 <- **lm**(y~x)  vb2[i] <- **coef**(mod7)[[2]]  }  mb2 <- **mean**(vb2)  seb2 <- **sd**(vb2) |

The mean and standard deviation of the estimated 40 values of *b*2 are, respectively, 9.974985 and 1.152632.

|  |
| --- |
| **plot**(**density**(vb2))  **curve**(**dnorm**(x, mb2, seb2), col="red", add=TRUE)  **legend**("topright", legend=**c**("true", "simulated"), lty=1, col=**c**("red", "black"))  **hist**(vb2, prob=TRUE, ylim=**c**(0,.4))  **curve**(**dnorm**(x, mean=mb2, sd=seb2), col="red", add=TRUE)  **rm**(list=**ls**()) *# Caution: this clears the Environment* |

The following figure shows the simulated distribution of *b*2 and the theoretical one.

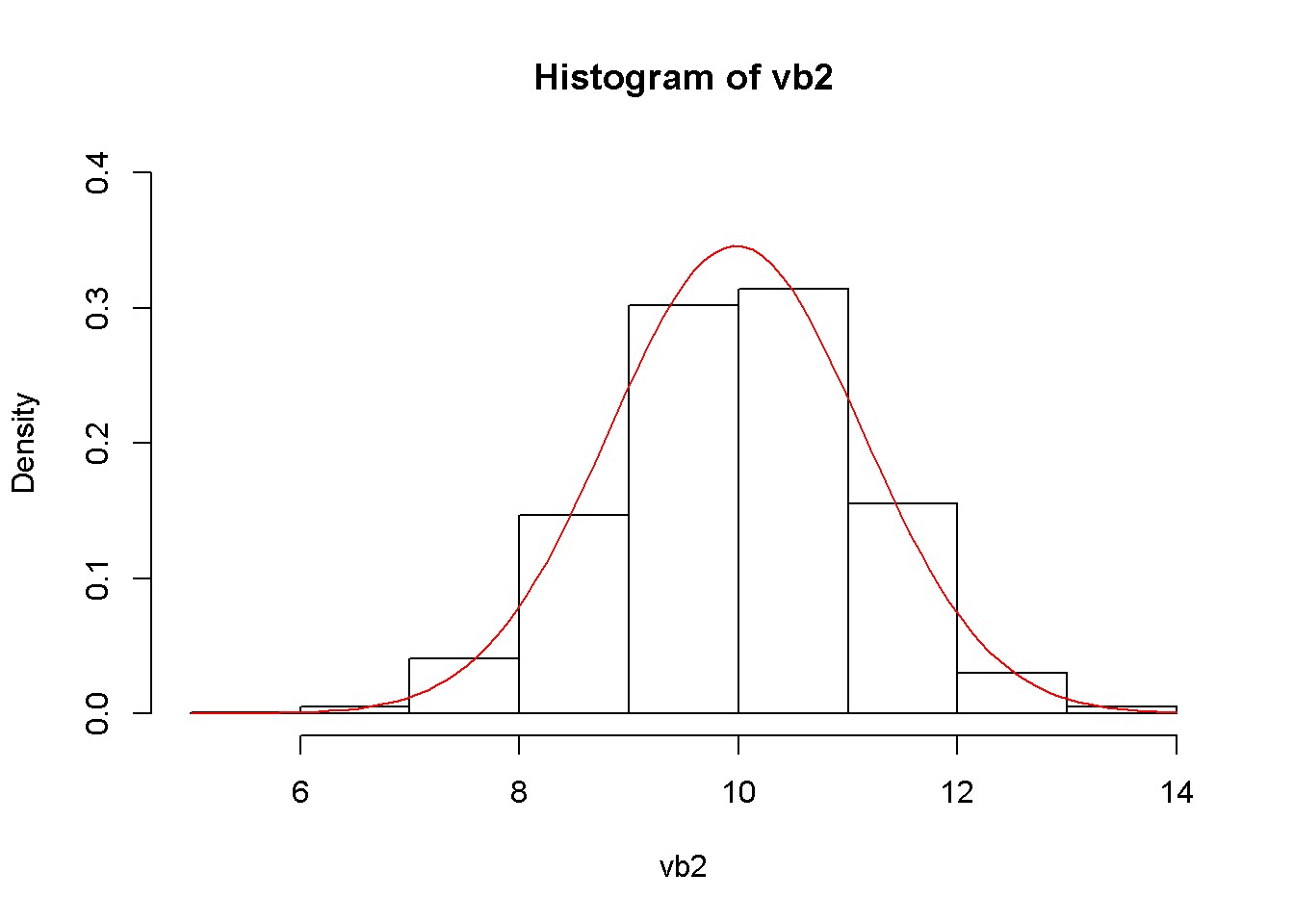
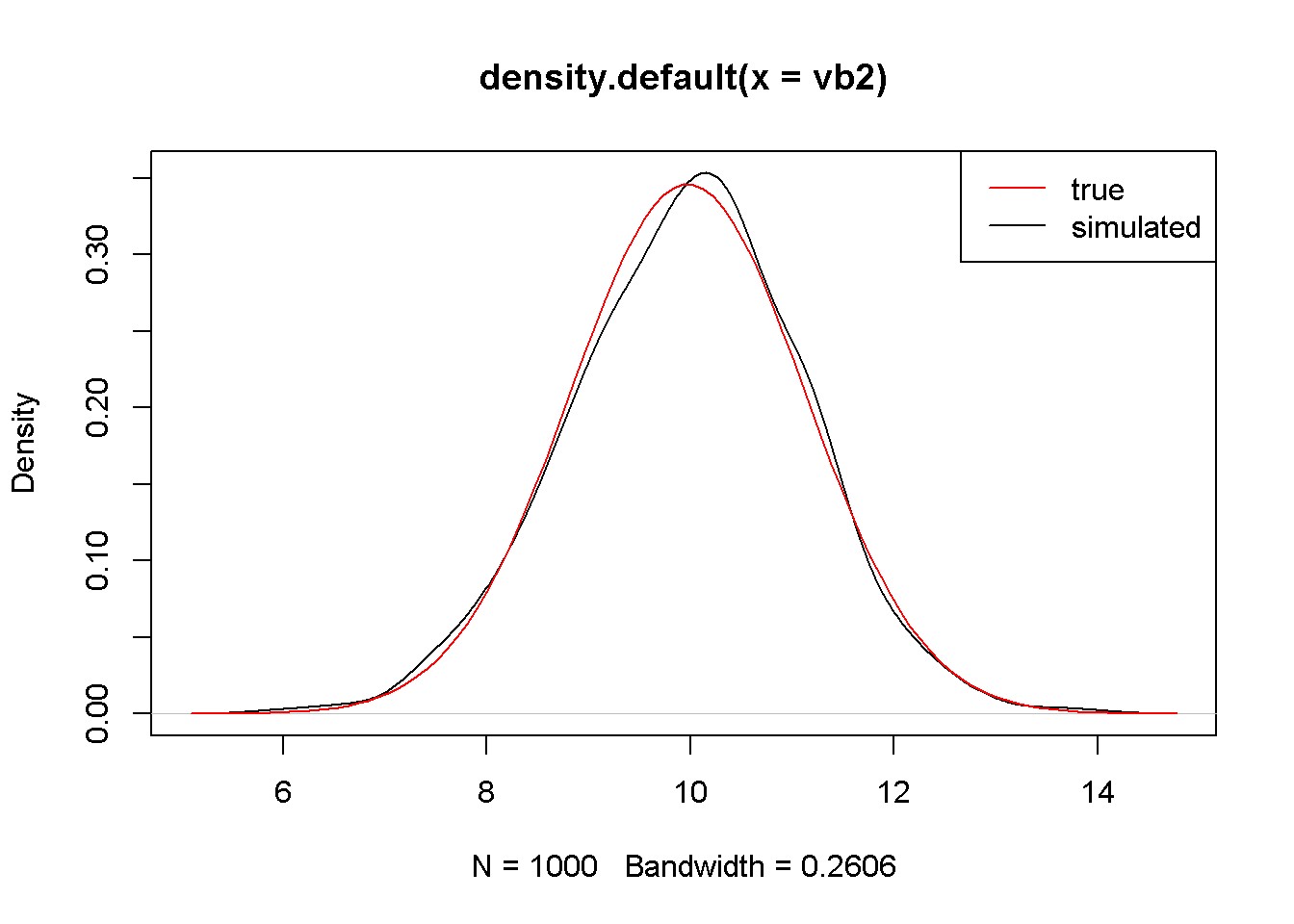


Figure: The simulated and theoretical distributions of *b*2

Video 2 -Interval Estimation

Understanding Interval Estimation:

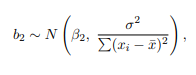
* Interval estimation involves finding a range (interval) within which we expect the true value of a parameter, like a regression coefficient, lies.
* Unlike a single estimate, an interval gives a range of plausible values, offering a better understanding of the estimate's precision.

**library**(xtable)

**library**(PoEdata)

**library**(knitr)

So far we estimated only a number for a regression parameter such as *β*2. This estimate gives no indication of its reliability, since it is just a realization of the random variable *b*2. An *interval estimate*, which is also known as a *confidence interval* is an interval centered on an estimated value, which includes the true parameter with a given probability, say 95%. A coefficient of the linear regression model such as *b*2 is **normally distributed** with its mean equal to the population parameter *β*2 and a variance that depends on the population variance *σ*2 and the sample size:



(1)

## The Estimated Distribution of Regression Coefficients

* In regression, coefficients (like b2) are not fixed numbers but random variables because they vary depending on the sample data.
* We often use the estimated coefficient (like b2 from our sample data) to guess the true population parameter (like β2).

Equation 1 gives the theoretical distribution of a linear regression coefficient, a distribution that is not very useful since it requires the unknown population variance *σ*2. If we replace *σ*2 with an estimated variance *σ*ˆ2 given in Equation 2, the standardized distribution of *b*2 becomes a *t* distribution with *N −* 2 degrees of freedom.

(2)

Equation 3 shows the the *t*-*ratio*:

(3)

## Confidence Interval in General

* A confidence interval includes the true parameter with a certain probability, such as 95%.
* It's a way of saying, "We are 95% confident that the true value lies within this range."
* The confidence interval for a coefficient is calculated using its estimated value, the standard error, and the t-distribution (since we typically don't know the population variance).
* The standard error measures the variability of the estimate.

An interval estimate of *b*2 based on the *t*-*ratio* is calculated in Equation 4, which we can consider as “an interval that includes the true parameter *β*2 with a probability of 100(1 *− α*)%.” In this context, *α* is called *significance level*, and the interval is called, for example, *a 95% confidence interval estimate for β*2. The critical value of the *t*-ratio, *tc*, depends on the chosen significance level and on the number of degrees of freedom. In *R*, the function that returns critical values for the *t* distribution is *qt(1 − ,df),* where *df* is the number of degrees of freedom.

 (4)

**A side note about using distributions in** *R***.** There are four types of functions related to distributions, each type’s name beginning with one of the following four letters: p for the cumulative distribution function, d for density, r for a draw of a random number from the respective distribution, and q for quantile. This first letter is followed by a few letters suggesting what distribution we refer to, such as norm, t, f, and chisq. Now, if we put together the first letter and the distribution name, we get functions such as the following, where x and q stand for quantiles, p stands for probability, df is degree of freedom (of which *F* has two), n is the desired number of draws, and lower.tail can be TRUE (default) if probabilities are *P*[*X ≤ x*] or FALSE if probabilities are *P*[*X > x*]:

• For the uniform distribution:

* + - dunif(x, min = 0, max = 1)
    - punif(q, min = 0, max = 1, lower.tail = TRUE)
    - qunif(p, min = 0, max = 1, lower.tail = TRUE)
    - runif(n, min = 0, max = 1)

• For the normal distribution:

* + - dnorm(x, mean = 0, sd = 1)
    - pnorm(q, mean = 0, sd = 1, lower.tail = TRUE)
    - qnorm(p, mean = 0, sd = 1, lower.tail = TRUE)
    - rnorm(n, mean = 0, sd = 1)

• For the *t* distribution:

* + - dt(x, df)
    - pt(q, df, lower.tail = TRUE)
    - qt(p, df, lower.tail = TRUE)
    - rt(n, df)

• For the *F* distribution:

* + - df(x, df1, df2)
    - pf(q, df1, df2, lower.tail = TRUE)
    - qf(p, df1, df2, lower.tail = TRUE)
    - rf(n, df1, df2)

• For the *χ*2 distribution:

* + - dchisq(x, df)
    - pchisq(q, df, lower.tail = TRUE) **–** qchisq(p, df, lower.tail = TRUE)
    - rchisq(n, df)

## Example: Confidence Intervals in the *food* Model

* Using R programming, you can calculate a confidence interval for a regression coefficient.
* For example, to find a 95% confidence interval for the coefficient on income in a food expenditure model, you would:
* Estimate the regression model (lm(food\_exp~income data=food)).
  + Calculate the standard error of the coefficient (seb2 <- coef(smod1)[22]).
  + Find the critical value from the t-distribution (tc <- qt(1-alpha/2, df)).
  + Calculate the lower and upper bounds of the interval (lowb <- b2 - tc\*seb2 and upb <- b2 + tc\*seb2).

Calculate a 95% confidence interval for the coefficient on *income* in the food expenditure model. Besides calculating confidence intervals, the following lines of code show how to retrieve information such as standard errors of coefficients from the summary() output. The function summary summarizes the results of a linear regression, some of which are not available directly from running the model itself.

|  |
| --- |
| **library**(PoEdata)  **data**("food")  alpha <- 0.05 *# chosen significance level*  mod1 <- **lm**(food\_exp~income, data=food)  b2 <- **coef**(mod1)[[2]]  df <- **df.residual**(mod1) *# degrees of freedom*  smod1 <- **summary**(mod1)  seb2 <- **coef**(smod1)[2,2] *# se(b2)*  tc <- **qt**(1-alpha/2, df)  lowb <- b2-tc\*seb2 *# lower bound*  upb <- b2+tc\*seb2 *# upper bound* |

The resulting confidence interval for the coefficient *b*2 in the *food* simple regression model is (5*.*97*,*14*.*45).

R has a special function*, confint(model)*, that can calculate confidence intervals taking as its argument the name of a regression model. The result of applying this function is a *K×*2 matrix with a confidence interval (two values: lower and upper bound) on each row and a number of lines equal to the number of parameters in the model (equal to 2 in the simple linear regression model). Compare the values from the next code to the ones from the previous to check that they are equal.

ci <- **confint**(mod1)

**print**(ci)

## 2.5 % 97.5 %

## (Intercept) -4.46328 171.2953 ## income 5.97205 14.4472

lowb\_b2 <- ci[2, 1] *# lower bound*

upb\_b2 <- ci[2, 2] *# upper bound.*

## Confidence Intervals in Repeated Samples

|  |
| --- |
| **data**("table2\_2")  alpha <- 0.05  mod1 <- **lm**(y1~x, data=table2\_2) *# just to determine df*  tc <- **qt**(1-alpha/2, df) *# critical t*  *# Initiate four vectors that will store the results:*  lowb1 <- **rep**(0, 10) *# 'repeat 0 ten times'*  upb1 <- **rep**(0, 10) *# (alternatively, 'numeric(10)')*  lowb2 <- **rep**(0, 10)  upb2 <-**rep**(0, 10)  *# One loop for each set of income:*  for(i in 2:11){ *# The curly bracket begins the loop*  dat <- **data.frame**(**cbind**(table2\_2[,1], table2\_2[,i]))  **names**(dat) <- **c**("x", "y")  mod1 <- **lm**(y~x, data=dat)  smod1 <- **summary**(mod1)  b1 <- **coef**(mod1)[[1]]  b2 <- **coef**(mod1)[[2]]  seb1 <- **coef**(smod1)[1,2]  seb2 <- **coef**(smod1)[2,2]  lowb1[i-1] <- b1-tc\*seb1  upb1[i-1] <- b1+tc\*seb1  lowb2[i-1] <- b2-tc\*seb2  upb2[i-1] <- b2+tc\*seb2  } *# This curly bracket ends the loop*  table <- **data.frame**(lowb1, upb1, lowb2, upb2)  **kable**(table, caption="Confidence intervals for $b\_{1}$ and $b\_{2}$", align="c") |

The Table shows the lower and upper bounds of the confidence intervals of *β*1 and *β*2.

Table: Confidence intervals for *b*1 and *b*2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| lowb1 | upb1 | lowb2 | upb2 | |
| 49.542182 | 213.846 | 2.51843 | 10.4413 | |
| -9.831097 | 124.323 | 7.64838 | 14.1174 | |
| 28.556681 | 179.264 | 4.50553 | 11.7727 | |
| -20.959444 | 113.968 | 8.64817 | 15.1545 | |
| 0.931168 | 167.534 | 5.27120 | 13.3049 | |
| -66.044847 | 119.302 | 9.08188 | 18.0194 | |
| -0.629753 | 129.046 | 7.80618 | 14.0592 | |
| 19.194721 | 140.129 | 6.84889 | 12.6804 | |
| 38.315701 | 156.287 | 5.20631 | 10.8950 | |
| 20.691744 | 171.232 | 4.13968 | 11.3988 | |
|  | | | | |

### Key Points:

* Intervals vs. Single Estimates: Confidence intervals provide more information than just a point estimate by showing a range where the true value likely falls.
* Significance Level (α): The choice of α (like 0.05 for a 95% confidence interval) affects the interval's width – lower α means a wider interval.
* Interpreting Confidence Intervals: If an interval for a coefficient does not include 0, it suggests the coefficient is significantly different from 0, indicating a potential relationship between the variables.

### Multiple Choice Questions:

1. What does an interval estimate in regression analysis typically provide?

A) A single fixed value of a parameter

B) A range within which the true parameter value is likely to fall

C) The exact value of the population variance

D) The probability of the sample mean being accurate

2. Why are regression coefficients considered random variables?

A) They change with each new sample

B) They are always equal to zero

C) They remain constant across different samples

D) They can only take integer values

3. A 95% confidence interval implies that:

A) The true parameter will always fall within this range

B) There's a 95% chance that the interval contains the true parameter

C) 95% of the sample data falls within this interval

D) The interval is 95 units wide

4. What is required to calculate a confidence interval for a regression coefficient?

A) Only the mean of the coefficient

B) The estimated coefficient and its standard error

C) The population variance

D) A z-distribution table

5. In the context of confidence intervals, what does the term 'α' represent?

A) Coefficient of determination

B) Significance level

C) Mean of the distribution

D) Degree of freedom

6. What does the function qt(1-alpha/2, df) in R return?

A) The mean of the t-distribution

B) The critical value from the t-distribution

C) The standard error of the estimate

D) The p-value for the test

7. What would be an indication that a regression coefficient is statistically significant?

A) Its confidence interval includes 0

B) Its confidence interval does not include 0

C) The interval is very narrow

D) The standard error is zero

Answer Key:

B) A range within which the true parameter value is likely to fall

A) They change with each new sample

B) There's a 95% chance that the interval contains the true parameter

B) The estimated coefficient and its standard error

B) Significance level

B) The critical value from the t-distribution

B) Its confidence interval does not include 0

Lab 1 – Hypothesis Test, P-Value and Testing Linear Combination of Parameters

## Hypothesis Tests

Hypothesis testing seeks to establish whether the data sample at hand provides sufficient evidence to support a certain conjecture (hypothesis) about a population parameter such as the intercept in a regression model, the slope, or some combination of them. The procedure requires three elements: the hypotheses (the *null* and the *alternative*), a test statistic, which in the case of the simple linear regression parameters is the *t*-ratio, and a significance level, *α*.

Suppose we believe that there is a significant relationship between a household’s income and its expenditure on food, a conjecture which has led us to formulate the food expenditure model in the first place. Thus, we believe that *β*2, the (population) parameter, is different from zero. Equation 5 shows the null and alternative hypotheses for such a test.

(5)

In general, if a null hypothesis *H*0 : *βk* = *c* is true, the *t* statistic (the *t*-ratio) is given by Equation 6 and has a *t* distribution with *N −* 2 degrees of freedom.

(6)

Test the hypothesis in Equation 5, which makes *c* = 0 in Equation 6. Let *α* = 0*.*05. The following Table shows the regression output.

Table: Regression output showing the coefficients

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 83.4160 | 43.41016 | 1.92158 | 0.062182 |
| income | 10.2096 | 2.09326 | 4.87738 | 0.000019 |

|  |
| --- |
| alpha <- 0.05  **library**(PoEdata); **library**(xtable); **library**(knitr)  **data**("food")  mod1 <- **lm**(food\_exp~income, data=food) smod1 <- **summary**(mod1)  table <- **data.frame**(**xtable**(mod1))  **kable**(table, caption="Regression output showing the coefficients")  b2 <- **coef**(mod1)[["income"]] *#coefficient on income*  *# or:*  b2 <- **coef**(mod1)[[2]] *# the coefficient on income*  seb2 <- **sqrt**(**vcov**(mod1)[2,2]) *#standard error of b2*  df <- **df.residual**(mod1) *# degrees of freedom*  t <- b2/seb2  tcr <- **qt**(1-alpha/2, df) |

The results *t* = 4*.*88 and *tcr* = 2*.*02 show that *t > tcr*, and therefore *t* falls in the rejection region (see the following Figure).

*# Plot the density function and the values of t:* **curve**(**dt**(x, df), -2.5\*seb2, 2.5\*seb2, ylab=" ", xlab="t") **abline**(v=**c**(-tcr, tcr, t), col=**c**("red", "red", "blue"),

lty=**c**(2,2,3))

**legend**("topleft", legend=**c**("-tcr", "tcr", "t"), col= **c**("red", "red", "blue"), lty=**c**(2, 2, 3))

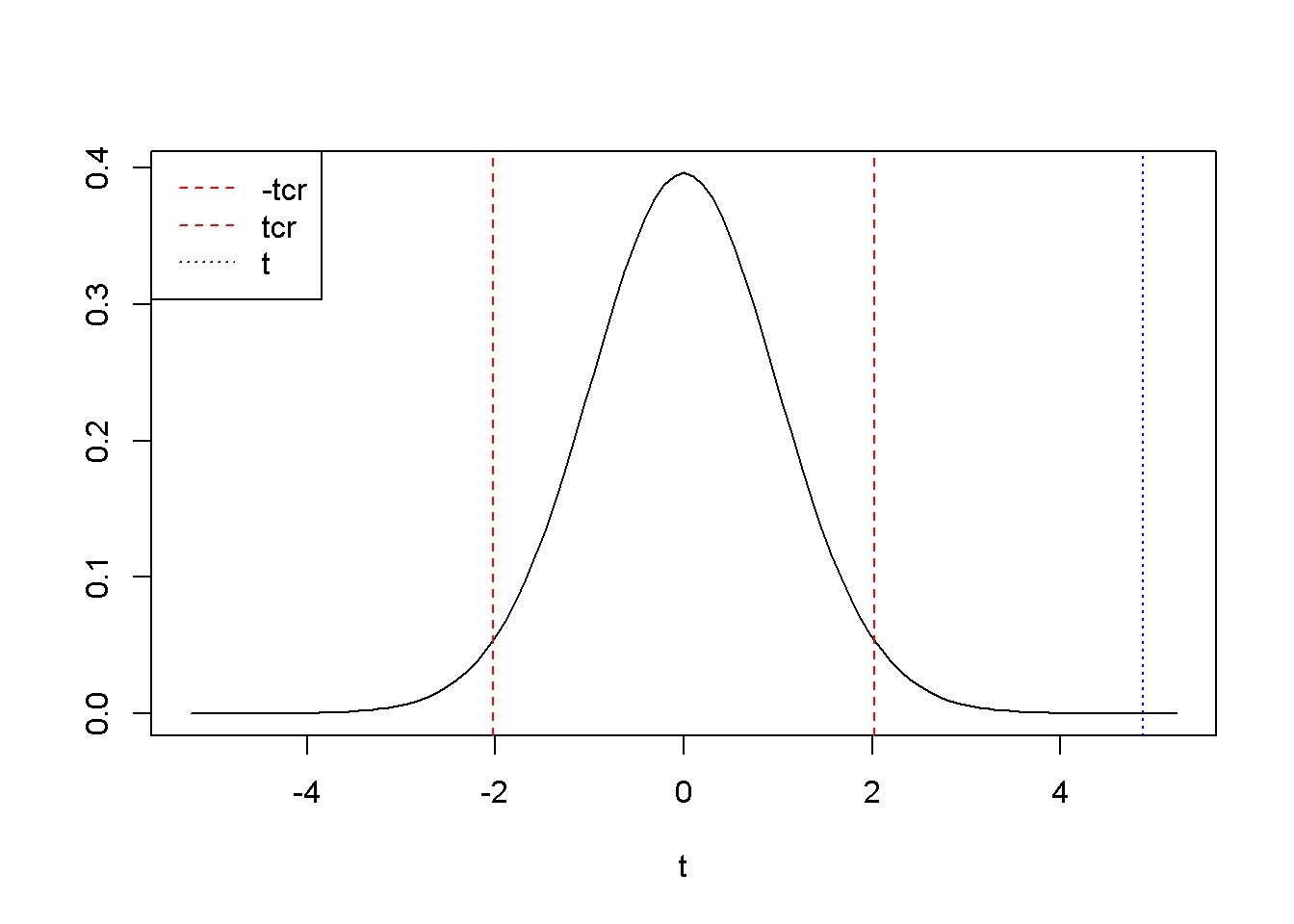


Figure: A two-tail hypothesis testing for *b*2 in the *food* example

Suppose we are interested to determine if *β*2 is greater than 5.5. This conjecture will go into the alternative hypothesis: *H*0 *≤* 5*.*5*, HA >* 5*.*5. The procedure is the same as for the two-tail test, but now the whole rejection region is to the right of the critical value *tcr*.

|  |
| --- |
| c <- 5.5  alpha <- 0.05  t <- (b2-c)/seb2  tcr <- **qt**(1-alpha, df) *# note: alpha is not divided by 2* **curve**(**dt**(x, df), -2.5\*seb2, 2.5\*seb2, ylab=" ", xlab="t") **abline**(v=**c**(tcr, t), col=**c**("red", "blue"), lty=**c**(2, 3)) **legend**("topleft", legend=**c**("tcr", "t"), col=**c**("red", "blue"), lty=**c**(2, 3)) |

The Figure below shows *tcr* = 1*.*685954, *t* = 2*.*249904. Since *t* falls again in the rejection region, we can reject the null hypothesis *H*0 : *β*2 *≤* 0.

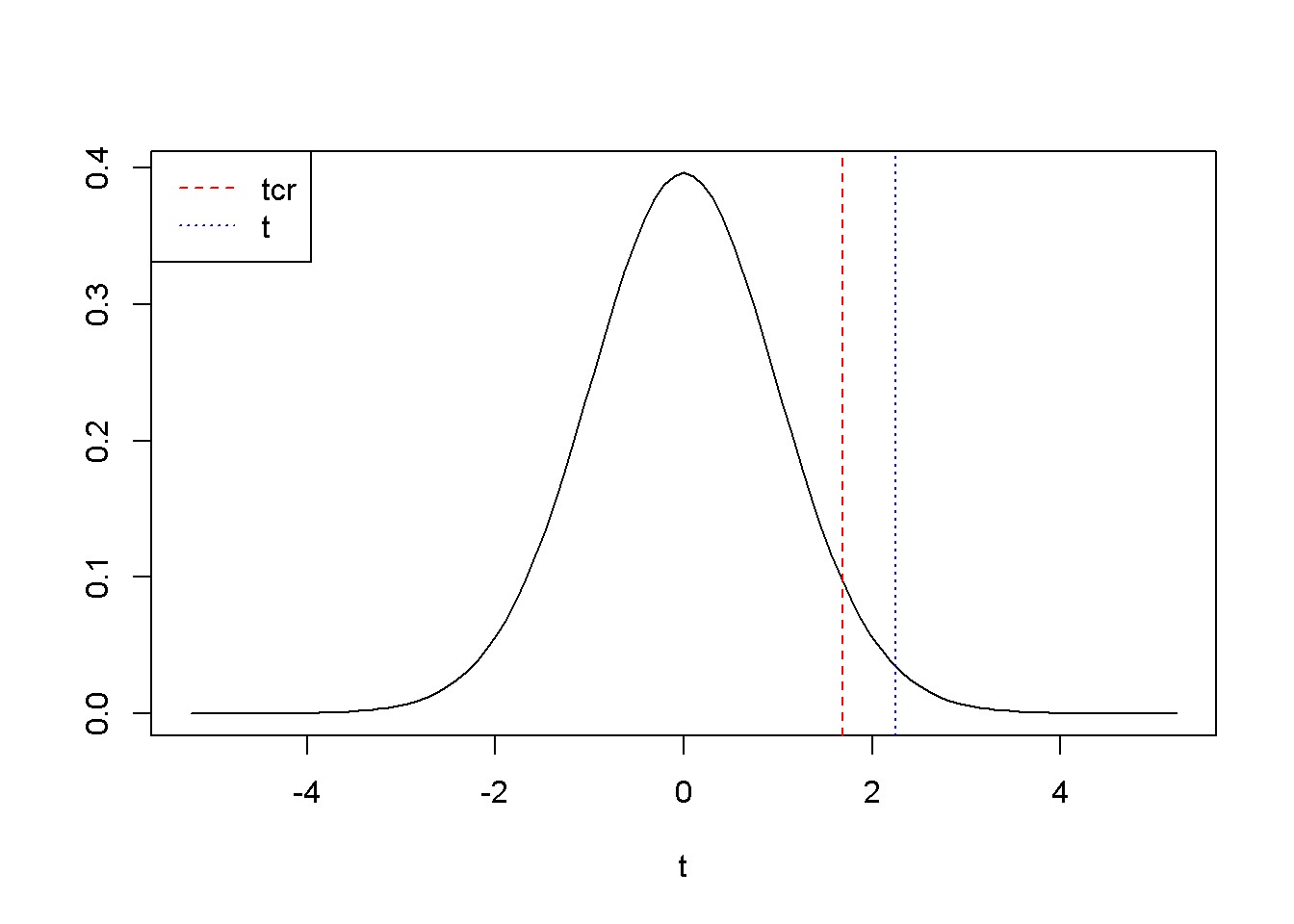


Figure: Right-tail test: the rejection region is to the right of *tcr*

A left-tail test is not different from the right-tail one, but of course the rejection region is to the left of *tcr*. For example, if we are interested to determine if *β*2 is less than15, we place this conjecture in the alternative hypothesis: *H*0 *≥* 15*, HA <* 15. The novelty here is how we use the qt() function to calculate *tcr*: instead of qt(1-alpha, ...), we need to use qt(alpha, ...). The Figure below illustrates this example, where the rejection region is, remember, to the left of *tcr*.

|  |
| --- |
| c <- 15 alpha <- 0.05 t <- (b2-c)/seb2  tcr <- **qt**(alpha, df) *# note: alpha is not divided by 2* **curve**(**dt**(x, df), -2.5\*seb2, 2.5\*seb2, ylab=" ", xlab="t") **abline**(v=**c**(tcr, t), col=**c**("red", "blue"), lty=**c**(2, 3)) **legend**("topleft", legend=**c**("tcr", "t"), col=**c**("red", "blue"), lty=**c**(2, 3)) |

R does automatically a *test of significance*, which is indeed testing the hypothesis *H*0 : *β*2 = 0*, HA* : *β*2 *6*= 0. The regression output shows the values of the *t*-ratio for all the regression coefficients.

**library**(PoEdata) **data**("food")

mod1 <- **lm**(food\_exp ~ income, data = food) table <- **data.frame**(**round**(**xtable**(**summary**(mod1)), 3)) **kable**(table, caption = "Regression output for the 'food' model")

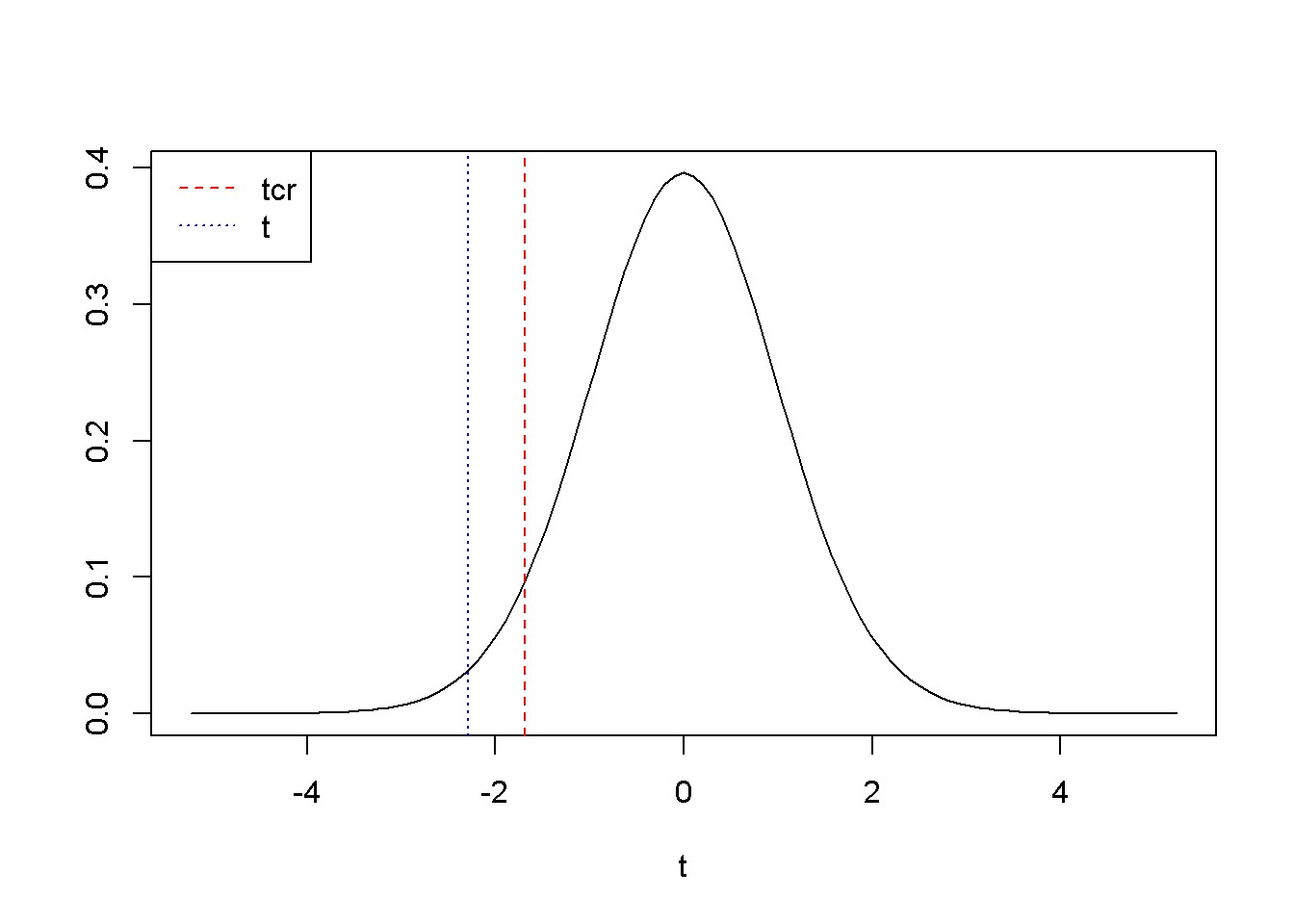


Figure: Left-tail test: the rejection region is to the left of *tcr* =

The Table below shows the regression output where the *t*-statistics of the coefficients can be observed.

Table: Regression output for the ’food’ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 83.416 | 43.410 | 1.922 | 0.062 |
| income | 10.210 | 2.093 | 4.877 | 0.000 |

Lab 2 -The *p*-Value

In the context of a hypothesis test, the *p*-value is the area outside the calculated *t*-statistic; it is the probability that the *t*-ratio takes a value that is more extreme than the calculated one, under the assumption that the null hypothesis is true. We reject the null hypothesis if the *p*-value is less than a chosen significance level. For a right-tail test, the *p*-value is the area to the right of the calculated *t*; for a left-tail test it is the area to the left of the calculated *t*; for a two-tail test the *p*-value is split in two equal amounts: *p/*2 to the left and *p/*2 to the right. *p*-values are calculated in *R* by the function pt(t, df), where *t* is the calculated *t*-ratio and *df* is the number of degrees of freedom in the estimated model.

Right-tail test, *H*0 : *β*2 *≤ c, HA* : *β*2 *> c*.

*# Calculating the p-value for a right-tail test*

c <- 5.5

t <- (b2-c)/seb2

p <- 1-**pt**(t, df) *# pt() returns p-values;*

The right-tail test shown in the previous lab gives the *p*-value *p* = 0*.*01516.

Left-tail test, *H*0 : *β*2 *≥ c, HA* : *β*2 *< c*.

|  |
| --- |
| *# Calculating the p-value for a left-tail test*  c <- 15  t <- (b2-c)/seb2  p <- **pt**(t, df) |

The left-tail test shown in the previous Lab gives the *p*-value *p* = 0*.*01388.

Two-tail test, *H*0 : *β*2 = *c, HA* : *β*2 ≠*c*.

|  |
| --- |
| *# Calculating the p-value for a two-tail test*  c <- 0  t <- (b2-c)/seb2  p <- 2\*(1-**pt**(**abs**(t), df)) |

The two-tail test shown in the Figure below gives the *p*-value *p* = 2 *×* 10*−*5, for a *t*-ratio *t* = 4*.*88.

**curve**(**dt**(x, df), from=-2.5\*seb2, to=2.5\*seb2)

**abline**(v=**c**(-t, t), col=**c**("blue", "blue"), lty=**c**(2, 2))

**legend**("topright", legend=**c**("-t", "t"), col=c("blue", "blue"), lty=c(2, 4))

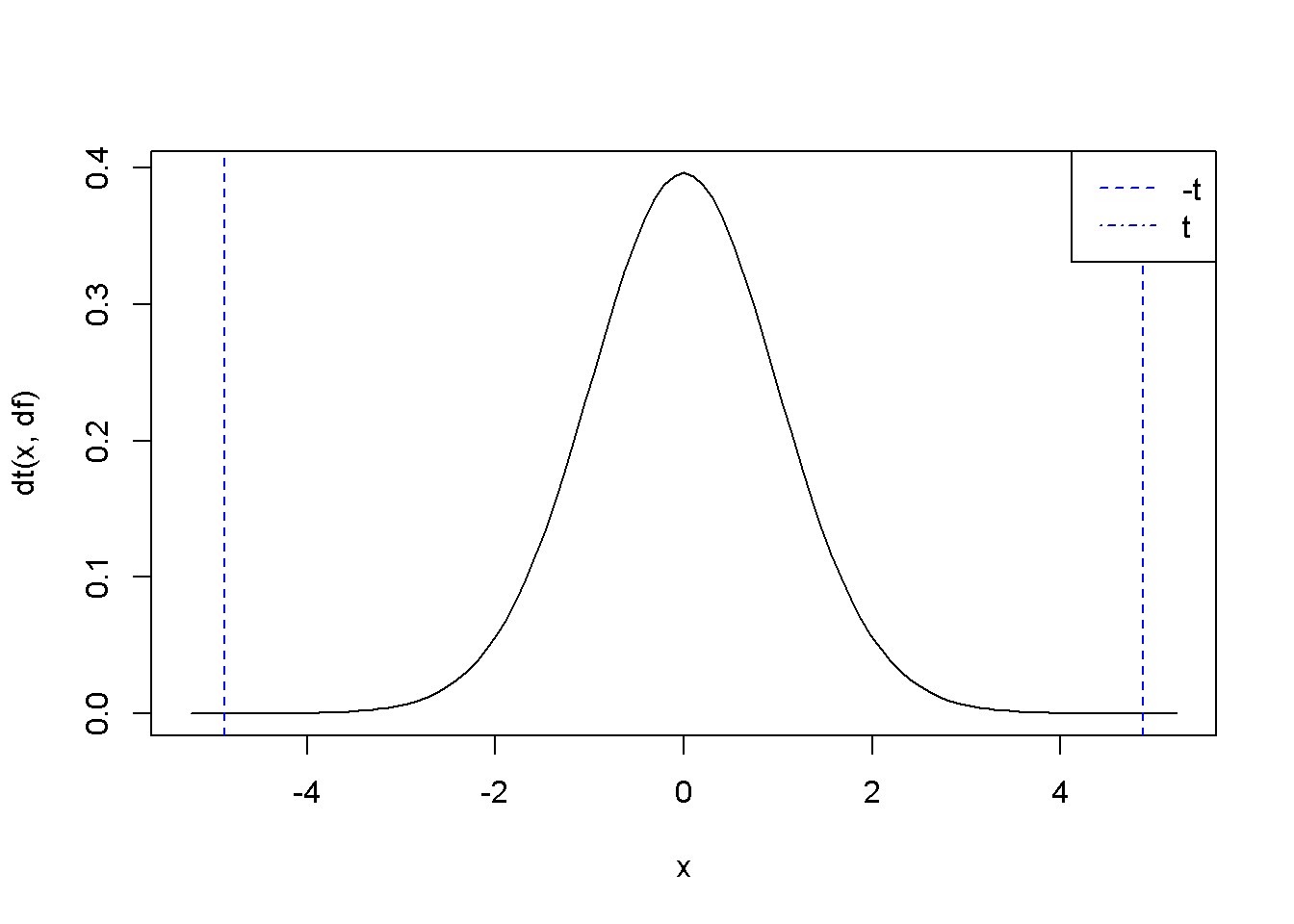


Figure: The *p*-value in two-tail hypothesis testing

R gives the *p*-values in the standard regression output, which we can retrieve using the summary(model) function. The Table below shows the output of the regression model, where the *p*-values can be observed.

|  |
| --- |
| table <- **data.frame**(**xtable**(smod1)) knitr::**kable**(table, caption=  "Regression output showing p-values") |

Table 3.4: Regression output showing p-values

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 83.4160 | 43.41016 | 1.92158 | 0.062182 |
| income | 10.2096 | 2.09326 | 4.87738 | 0.000019 |

## Testing Linear Combinations of Parameters

Sometimes we wish to estimate the expected value of the dependent variable, *y*, for a given value of *x*. For example, according to our *food* model, what is the average expenditure of a household having income of $2000? We need to estimate the linear combination of the regression coefficients *β*1 and *β*2 given in Equation 3.7 (let’s denote the linear combination by *L*).

*L* = *E*(*food*\_*exp|income* = 20) = *β*1 + 20*β*2 (3.7)

Finding confidence intervals and testing hypotheses about the linear combination in Equation 3.7 requires calculating a *t*-statistic similar to the one for the regression coefficients we calculated before. However, estimating the standard error of the linear combination is not as straightforward. In general, if *X* and *Y* are two random variables and *a* and *b* two constants, the variance of the linear combination *aX* + *bY* is

*var*(*aX* + *bY* ) = *a*2*var*(*X*) + *b*2*var*(*Y* ) + 2*abcov*(*X,Y* )*.* (3.8)

Now, let us apply the formula in Equation 3.8 to the linear combination of *β*1 and *β*2 given by Equation 3.7, we obtain Equation 3.9.

*var*(*b*1 + 20*b*2) = *var*(*b*1) + 202*var*(*b*2) + 2 *×* 20*cov*(*b*1*b*2) (3.9)

The following sequence of code determines an interval estimate for the expected value of food expenditure in a household earning $2000 a week.

|  |
| --- |
| **library**(PoEdata) **data**("food") alpha <- 0.05 x <- 20 *# income is in 100s, remember?* m1 <- **lm**(food\_exp~income, data=food) tcr <- **qt**(1-alpha/2, df) *# rejection region right of tcr.*  df <- **df.residual**(m1) b1 <- m1$coef[1] b2 <- m1$coef[2] varb1 <- **vcov**(m1)[1, 1] varb2 <- **vcov**(m1)[2, 2] covb1b2 <- **vcov**(m1)[1, 2] L <- b1+b2\*x *# estimated L*  varL = varb1 + x^2 \* varb2 + 2\*x\*covb1b2 *# var(L)* seL <- **sqrt**(varL) *# standard error of L* lowbL <- L-tcr\*seL upbL <- L+tcr\*seL |

*3.7. TESTING LINEAR COMBINATIONS OF PARAMETERS*

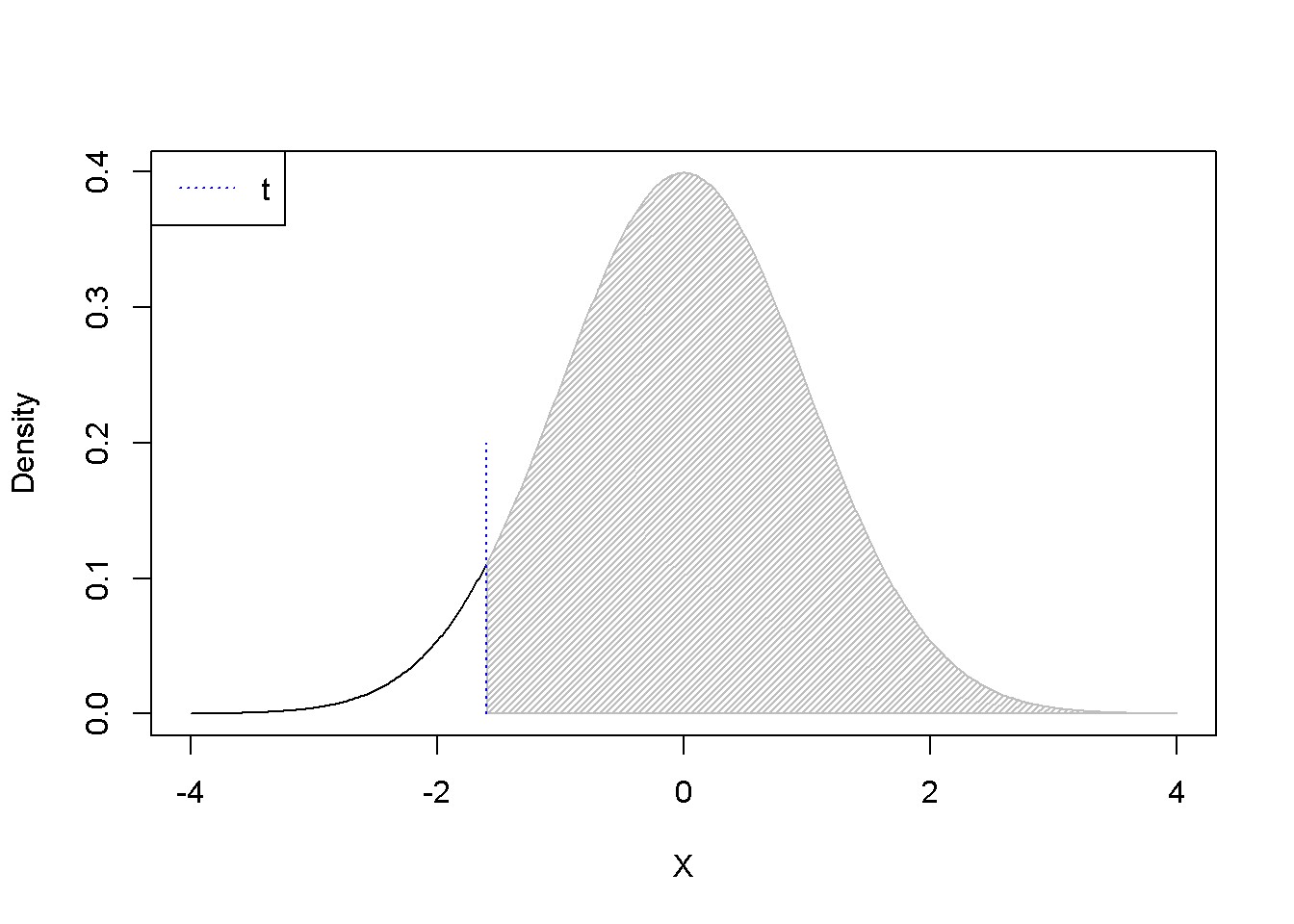
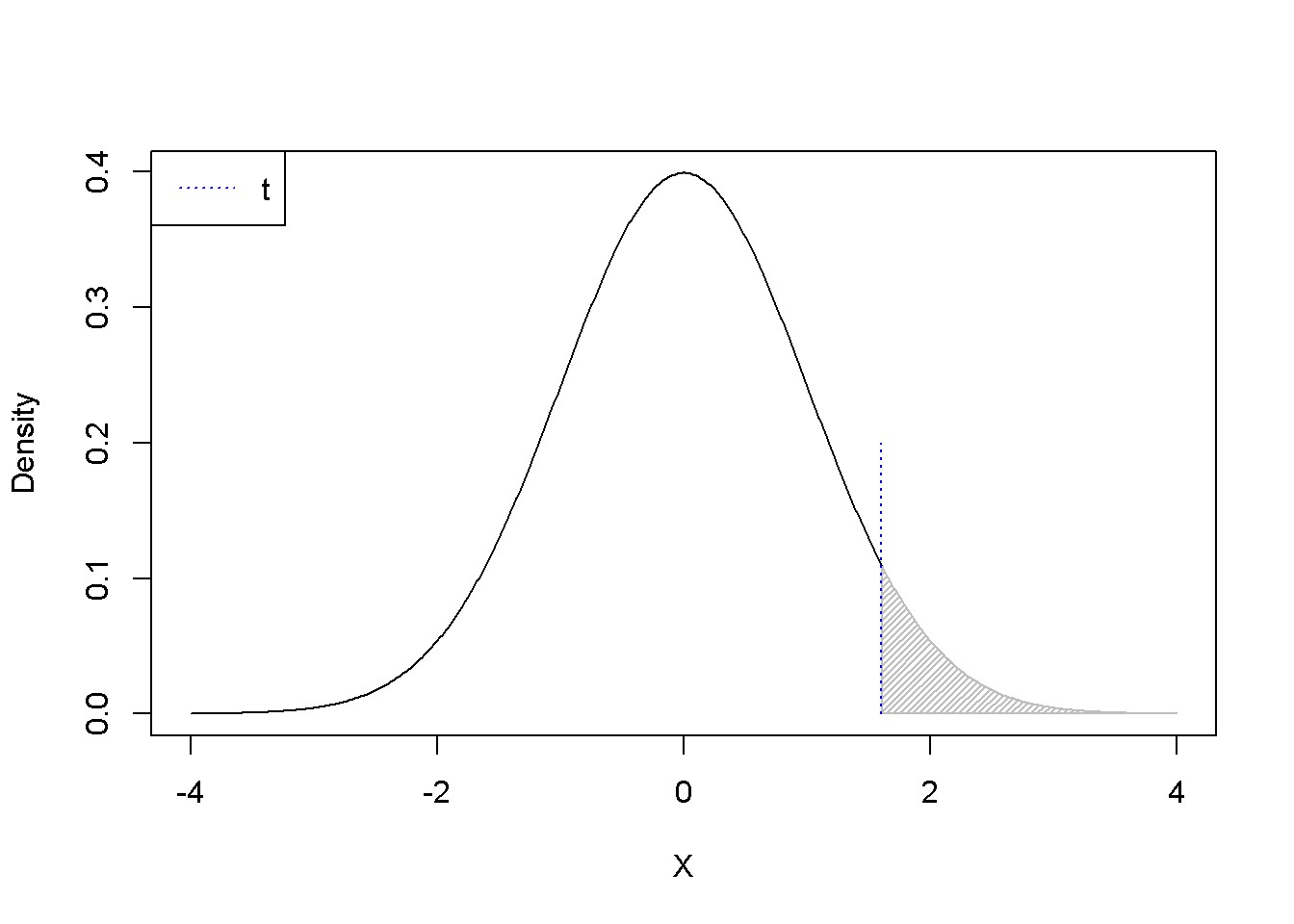


Figure 3.5: *p−*Values for positive and negative *t* as calculated using the formula 1 *− pt*(*t,df*)

The result is the confidence interval (258*.*91, 316*.*31). Next, we test hypotheses about the linear combination *L* defined in Equation 3.7, looking at the three types of hypotheses: two-tail, left-tail, and right-tail. Equations 3.10 *−* 3.12 show the test setups for a hypothesized value of food expenditure *c*.

|  |  |  |
| --- | --- | --- |
| *H*0 : *L* = *c,* | *HA* : *L 6*= *c* | (3.10) |
| *H*0 : *L ≥ c,* | *HA* : *L < c* | (3.11) |
| *H*0 : *L ≤ c,* | *HA* : *L > c* | (3.12) |

One should use the function pt(t, df) carefully, because it gives wrong results when testing hypotheses using the *p*-value metod and the calculated *t* is negative. Therefore, the absolute value of *t* should be used. Figure 3.5 shows the *p*-values calculated with the formula 1-pt(t, df). When *t* is positive and the test is two-tail, doubling the *p*-value 1-pt(t, df) is correct; but when *t* is negative, the correct *p*-value is 2\*p(t, df).

**.shadenorm**(above=1.6, justabove=TRUE) **segments**(1.6,0,1.6,0.2,col="blue", lty=3)

**legend**("topleft", legend="t", col="blue", lty=3)

**.shadenorm**(above=-1.6, justabove=TRUE) **segments**(-1.6,0,-1.6,0.2,col="blue", lty=3) **legend**("topleft", legend="t", col="blue", lty=3) The next sequence uses the values already calculated before, a hypothesized level of food expenditure *c*=$250, and an income of $2000; it tests the two-tail hypothesis in Equation 3.10 first using the “critical *t*” method, then using the *p*-value method.

|  |
| --- |
| c <- 250 alpha <- 0.05  t <- (L-c)/seL *# t < tcr --> Reject Ho.*  tcr <- **qt**(1-alpha/2, df)  *# Or, we can calculate the p-value, as follows:*  p\_value <- 2\*(1-**pt**(**abs**(t), df)) *#p<alpha -> Reject Ho* |

The results are: *t* = 2*.*65, *tcr* = 2*.*02, and *p* = 0*.*0116. Since *t > tcr*, we reject the null hypothesis. The same result is given by the *p*-value method, where the *p*-value is twice the probability area determined by the calculated *t*.

Lab 3 - Prediction, R-squared, and Modelling

**rm**(list=**ls**()) *# Caution: this clears the Environment*

A **prediction** is an estimate of the value of *y* for a given value of *x*, based on a regression model of the form shown in Equation 4.1. **Goodness-of-fit** is a measure of how well an estimated regression line approximates the data in a given sample. One such measure is the correlation coefficient between the predicted values of *y* for all *x*-s in the data file and the actual *y*-s. Goodness-of-fit, along with other diagnostic tests help determining the most suitable **functional form** of our regression equation, i.e., the most suitable mathematical relationship between *y* and *x*.

*yi* = *β*1 + *β*2*xi* + *ei* (4.1)

## Forecasting (Predicting a Particular Value)

Assuming that the expected values of the error term in Equation 4.1 is zero, Equation 4.2 gives *y*ˆ*i*, the predicted value of the expectation of *yi* given *xi*, where *b*1 and *b*2 are the (least squares) estimates of the regression parameters *β*1 and *β*2.

*y*ˆ*i* = *b*1 + *b*2*xi* (4.2)

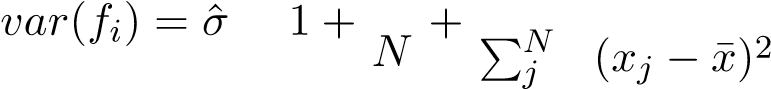
The predicted value *y*ˆ*i* is a random variable, since it depends on the sample; therefore, we can calculate a confidence interval and test hypothesis about it, provided we can determine its distribution and variance. The prediction has a normal distribution,

45

being a linear combination of two normally distributed random variables *b*1 and *b*2, and its variance is given by Equation 4.3. Please note that the variance in Equation 4.3 is not the same as the one in Equation 3.9; the former is the variance of the estimated *expectation* of *y*, while the latter is the variance of a particular occurrence of *y*. Let us call the latter the variance of the forecast error. Not surprisingly, the variance of the forecast error is greater than the variance of the predicted *E*(*y|x*).

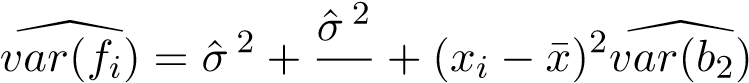
As before, since we need to use an estimated variance, we use a *t*-distribution instead of a normal one. Equation 4.3 applies to any given *x*, say *x*0, not only to those *x*-s in the dataset.

" 2 #

 \ 2 1 (*xi − x*¯) *,* (4.3)

=1

which can be reduced to

 (4.4)

*N*

Let’s determine a standard error for the *food* equation for a household earning $2000 a week, i.e., at *x* = *x*0 = 20, using Equation 4.4; to do so, we need to retrieve *var*(*b*2) and *σ*ˆ, the standard error of regression from the regression output.

|  |
| --- |
| **library**(PoEdata) **data**("food") alpha <- 0.05 x <- 20  xbar <- **mean**(food$income) m1 <- **lm**(food\_exp~income, data=food) b1 <- **coef**(m1)[[1]] b2 <- **coef**(m1)[[2]] yhatx <- b1+b2\*x sm1 <- **summary**(m1) df <- **df.residual**(m1) tcr <- **qt**(1-alpha/2, df)  N <- **nobs**(m1) *#number of observations, N* N <- **NROW**(food) *#just another way of finding N* varb2 <- **vcov**(m1)[2, 2]  sighat2 <- sm1$sigma^2 *# estimated variance*  varf <- sighat2+sighat2/N+(x-xbar)^2\*varb2 *#forecast variance* sef <- **sqrt**(varf) *#standard error of forecast* lb <- yhatx-tcr\*sef ub <- yhatx+tcr\*sef |

*4.1. FORECASTING (PREDICTING A PARTICULAR VALUE)*

The result is the confidence interval for the forecast (104*.*13*,*471*.*09), which is, as expected, larger than the confidence interval of the estimated expected value of *y* based on Equation 3.9.

Let us calculate confidence intervals of the forecast for all the observations in the sample and draw the upper and lower limits together with the regression line. Figure 4.1 shows the confidence interval band about the regression line.

|  |
| --- |
| sef <- **sqrt**(sighat2+sighat2/N+(food$income-xbar)^2\*varb2) yhatv <- **fitted.values**(m1) lbv <- yhatv-tcr\*sef ubv <- yhatv+tcr\*sef xincome <- food$income  dplot <- **data.frame**(xincome, yhatv, lbv, ubv) dplotord <- dplot[**order**(xincome), ] xmax <- **max**(dplotord$xincome) xmin <- **min**(dplotord$xincome) ymax <- **max**(dplotord$ubv) ymin <- **min**(dplotord$lbv)  **plot**(dplotord$xincome, dplotord$yhatv, xlim=**c**(xmin, xmax), ylim=**c**(ymin, ymax),  xlab="income", ylab="food expenditure", type="l")  **lines**(dplotord$ubv~dplotord$xincome, lty=2) **lines**(dplotord$lbv~dplotord$xincome, lty=2) |

A different way of finding point and interval estimates for the *predicted E*(*y|x*) and *forecasted y* (please see the distinction I mentioned above) is to use the predict() function in R. This function requires that the values of the independent variable where the prediction (or forecast) is intended have a data frame structure. The next example shows in parallel point and interval estimates of *predicted* and *forecasted* food expenditures for income is $2000. As I have pointed out before, the point estimate is the same for both prediction and forecast, but the interval estimates are very different.

incomex=**data.frame**(income=20)

**predict**(m1, newdata=incomex, interval="confidence",level=0.95)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| fit | | | lwr | | upr 316.311 | |
| 287.609 | | | 258.907 | |
| **predict**(m1, newdata=incomex, interval="prediction",level=0.95) | | | | | | |
| fit | lwr | | upr | |
| 287.609 | 104.132 | | 471.085 | |

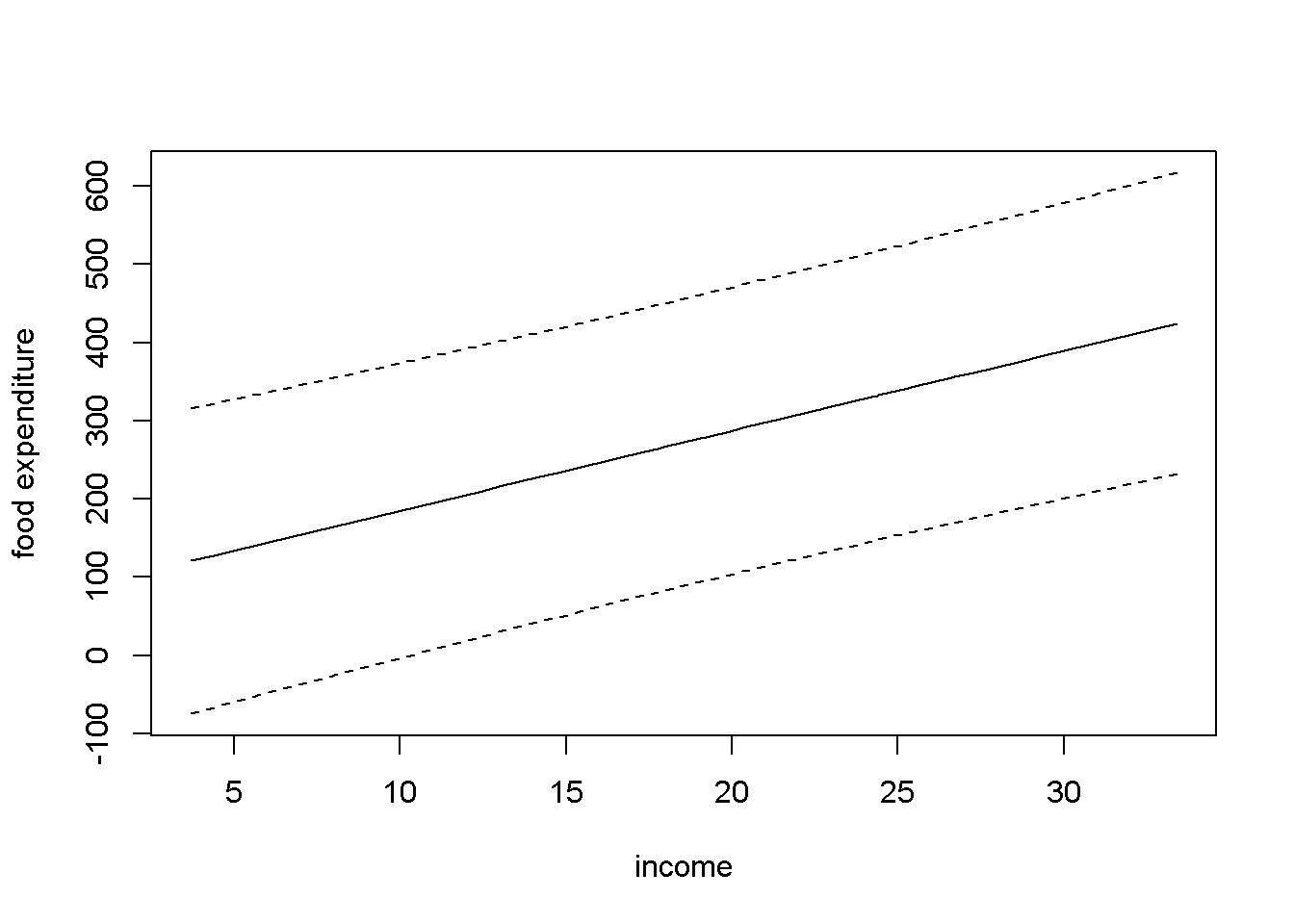


Figure 4.1: Forecast confidence intervals for the *food* simple regression

Let us now use the predict() function to replicate Figure 4.1. The result is Figure

4.2, which shows, besides the interval estimation band, the points in the dataset. (I will create new values for income just for the purpose of plotting.)

|  |
| --- |
| xmin <- **min**(food$income) xmax <- **max**(food$income) income <- **seq**(from=xmin, to=xmax)  ypredict <- **predict**(m1, newdata=**data.frame**(income), interval="confidence")  yforecast <- **predict**(m1, newdata=**data.frame**(income), interval="predict")  **matplot**(income, **cbind**(ypredict[,1], ypredict[,2], ypredict[,3], yforecast[,2], yforecast[,3]),  type ="l", lty=**c**(1, 2, 2, 3, 3),  col=**c**("black", "red", "red", "blue", "blue"), ylab="food expenditure", xlab="income")  **points**(food$income, food$food\_exp) **legend**("topleft", legend=**c**("E[y|x]", "lwr\_pred", "upr\_pred",  "lwr\_forcst","upr\_forcst"), lty=**c**(1, 2, 2, 3, 3),  col=**c**("black", "red", "red", "blue", "blue") ) |

*4.2. GOODNESS-OF-FIT*

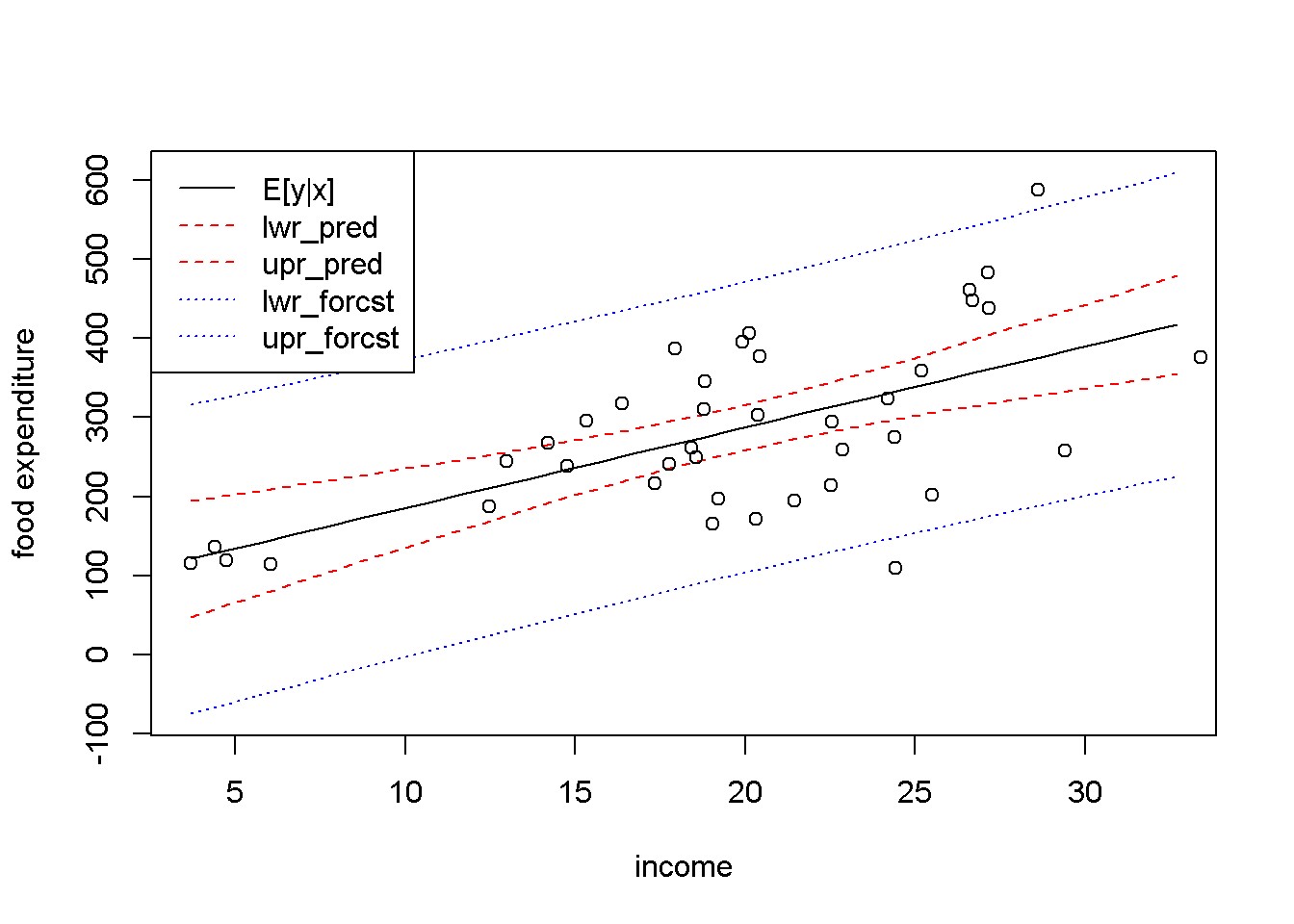


Figure 4.2: Predicted and forecasted bands for the *food* dataset

Figure 4.2 presents the predicted and forecasted bands on the same graph, to show that they have the same point estimates (the black, solid line) and that the forecasted band is much larger than the predicted one. Put another way, you may think about the distinction between the two types of intervals that we called *prediction* and *forecast* as follows: the prediction interval is not supposed to include, say, 95 percent of the points, but to include the regression line, *E*(*y|x*), with a probability of 95 percent; the forecasted interval, on the other hand, should include any true point with a 95 percent probability.

## 4.2 Goodness-of-Fit

The total variation of *y* about its sample mean, *SST*, can be decomposed in variation about the regression line, *SSE*, and variation of the regression line about the mean of *y*, *SSR*, as Equation 4.5 shows.

*SST* = *SSR* + *SSE* (4.5)

The **coefficient of determination**, *R*2, is defined as the proportion of the variance in *y* that is explained by the regression, *SSR*, in the total variation in *y*, *SST*. Dividing both sides of the Equation 4.5 by *SST* and re-arranging terms gives a formula to calculate *R*2, as shown in Equation 4.6.

2 *SSR SSE*

*R* = = 1 *−*  (4.6)

*SST SST*

*R*2 takes values between 0 and 1, with higher values showing a closer fit of the regression line to the data. In *R*, the value of *R*2 can be retrieved from the summary of the regression model under the name r.squared; for instance, in our *food* example,

*R*2 = 0*.*385. *R*2 is also printed as part of the summary of a regression model, as the following code sequence shows. (The parentheses around a command tells *R* to print the result.)

(rsq <- sm1$r.squared) *#or*

## [1] 0.385002

sm1 *#prints the summary of regression model m1*

##

## Call:

## lm(formula = food\_exp ~ income, data = food)

##

## Residuals:

## Min 1Q Median 3Q Max

## -223.03 -50.82 -6.32 67.88 212.04

##

## Coefficients:

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

## (Intercept) 83.42 43.41 1.92 0.062 .

## income 10.21 2.09 4.88 0.000019 \*\*\*

## ---

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

##

## Residual standard error: 89.5 on 38 degrees of freedom

## Multiple R-squared: 0.385, Adjusted R-squared: 0.369

## F-statistic: 23.8 on 1 and 38 DF, p-value: 0.0000195

If you need the sum of squared errors, *SSE*, or the sum of squares due to regression, *SSR*, use the anova function, which has the structure shown in Table 4.1.

anov <- **anova**(m1) dfr <- **data.frame**(anov)

**kable**(dfr,

caption="Output generated by the `anova` function")

Table 4.1 indicates that *SSE* = anov[2,2] = 3*.*045052 *×* 105, *SSR* = anov[1,2]

= 1*.*90627 *×* 105, and *SST* = anov[1,2]+anov[2,2] = 4*.*951322 *×* 105. In our

*4.3. LINEAR-LOG MODELS*

Table 4.1: Output generated by the ‘anova‘ function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum.Sq | Mean.Sq | F.value | Pr..F. |
| income | 1 | 190627 | 190626.98 | 23.7888 | 0.000019 |
| Residuals | 38 | 304505 | 8013.29 | NA | NA |

simple regression model, the sum of squares due to regression only includes the variable income. In multiple regression models, which are models with more than one independent variable, the sum of squares due to regression is equal to the sum of squares due to all independent variables. The anova results in Table 4.1 include other useful information: the number of degrees of freedom, anov[2,1] and the estimated variance *σ*ˆ 2 =anov[2,3].

Lab 4 Linear-Log Models

Non-linear functional forms of regression models are useful when the relationship between two variables seems to be more complex than the linear one. One can decide to use a non-linear functional form based on a mathematical model, reasoning, or simply inspecting a scatter plot of the data. In the *food expenditure* model, for example, it is reasonable to believe that the amount spent on food increases faster at lower incomes than at higher incomes. In other words, it increases at a decreasing rate, which makes the regression curve flatten out at higher incomes.

What function could one use to model such a relationship? The logarithmic function fits this profile and, as it turns out, it is relatively easy to interpret, which makes it very popular in econometric models. The general form of a *linear-log* econometric model is provided in Equation 4.7.

*yi* = *β*1 + *β*2*log*(*xi*) + *ei* (4.7)

The *marginal effect* of a change in *x* on *y* is the slope of the regression curve and is given by Equation 4.8; unlike in the linear form, it depends on *x* and it is, therefore, only valid for small changes in *x*.

*dy β*2

= (4.8) *dx x*

Related to the linear-log model, another measure of interest in economics is the *semi-elasticity* of *y* with respect to *x*, which is given by Equation 4.9. Semi-elasticity suggests that a change in *x* of 1% changes *y* by *β*2*/*100 units of *y*. Since semi-elasticity also changes when *x* changes, it should only be determined for small changes in *x*.

Table 4.2: Linear-log model output for the \*food\* example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | -97.1864 | 84.2374 | -1.15372 | 0.25582 |
| log(income) | 132.1658 | 28.8046 | 4.58836 | 0.00005 |

*dy* = (%∆*x*) (4.9)

Another quantity that might be of interest is the *elasticity* of *y* with respect to *x*, which is given by Equation 4.10 and indicates that a one percent increase in *x* produces a (*β*2*/y*) percent change in *y*.

*β*2

%∆*y* = (%∆*x*) (4.10)

*y*

Let us estimate a linear-log model for the *food* dataset, draw the regression curve, and calculate the marginal effects for some given values of the dependent variable.

mod2 <- **lm**(food\_exp~**log**(income), data=food)

tbl <- **data.frame**(**xtable**(mod2)) **kable**(tbl, digits=5, caption="Linear-log model output for the \*food\* example")

|  |
| --- |
| b1 <- **coef**(mod2)[[1]] b2 <- **coef**(mod2)[[2]]  pmod2 <- **predict**(mod2, newdata=**data.frame**(income), interval="confidence")  **plot**(food$income, food$food\_exp, xlab="income", ylab="food expenditure")  **lines**(pmod2[,1]~income, lty=1, col="black") **lines**(pmod2[,2]~income, lty=2, col="red") **lines**(pmod2[,3]~income, lty=2, col="red") |
| x <- 10 *#for a household earning #1000 per week* y <- b1+b2\***log**(x)  DyDx <- b2/x *#marginal effect* DyPDx <- b2/100 *#semi-elasticity*  PDyPDx <- b2/y *#elasticity* |

The results for an income of $1000 are as follows: *dy/dx* = 13*.*217, which indicates that an increase in income of $100 (i.e., one unit of *x*) increases expenditure by $

*4.4. RESIDUALS AND DIAGNOSTICS*

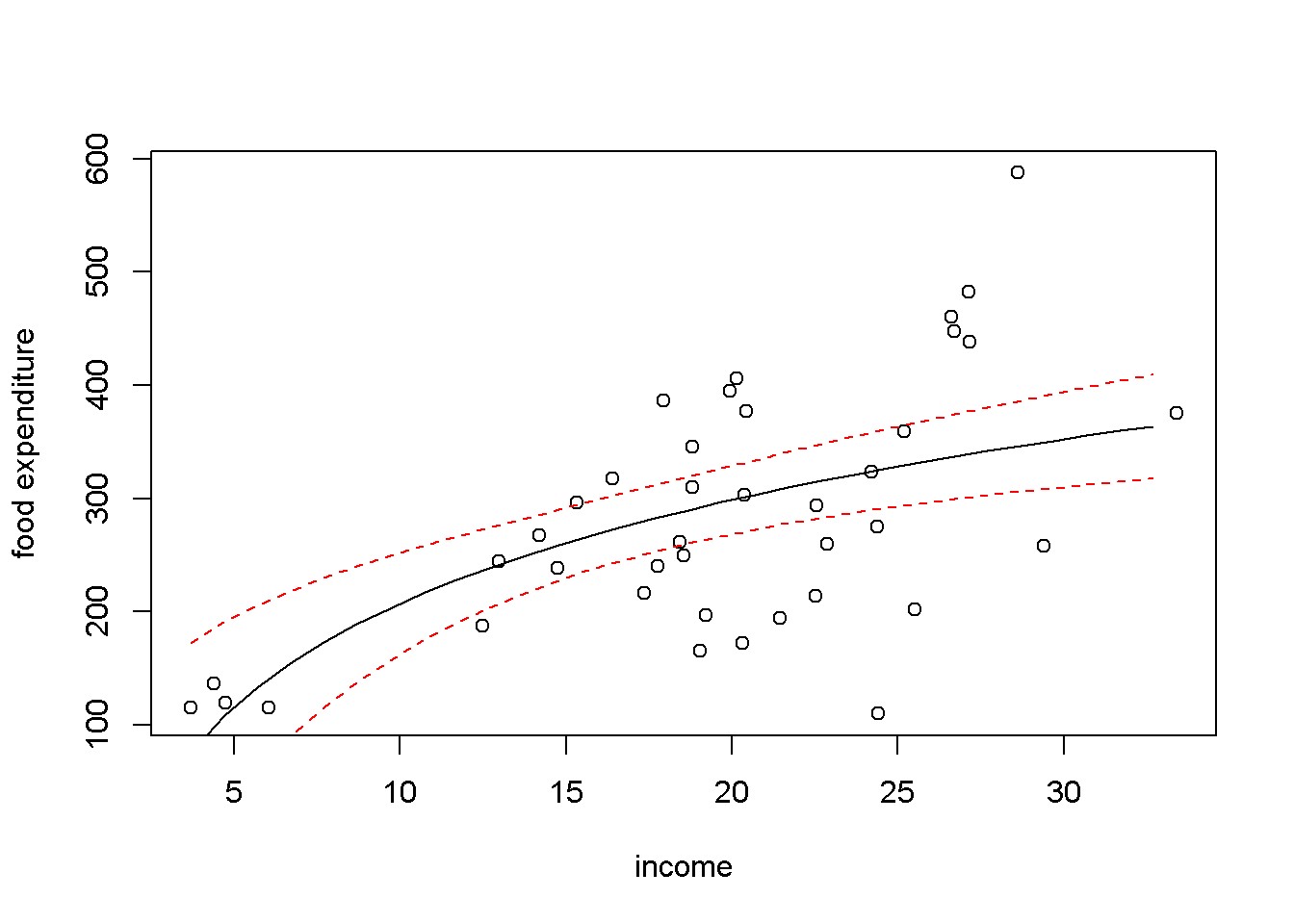


Figure 4.3: Linear-log representation for the *food* data

13*.*217; for a 1% increase in income, that is, an increase of $10, expenditure increases by $ 1*.*322; and, finally, for a 1% increase in income expenditure incrases by 0*.*638%.

## 4.4 Residuals and Diagnostics

Regression results are reliable only to the extent to which the underlying assumptions are met. Plotting the residuals and calculating certain test statistics help deciding whether assumptions such as homoskedasticity, serial correlation, and normality of the errors are not violated. In R, the residuals are stored in the vector residuals of the regression output.

ehat <- mod2$residuals **plot**(food$income, ehat, xlab="income", ylab="residuals")

Figure 4.4 shows the residuals of the of the linear-log equation of the *food expenditure* example. One can notice that the spread of the residuals seems to be higher at higher incomes, which may indicate that the heteroskedasticity assumption is violated.

Let us draw a residual plot generated with a simulated model that satisfies the regression assumptions. The data generating process is given by Equation 4.11, where *x* is a number between 0 and 10, randomly drawn from a uniform distribution, and the error term is randomly drawn from a standard normal distribution. Figure

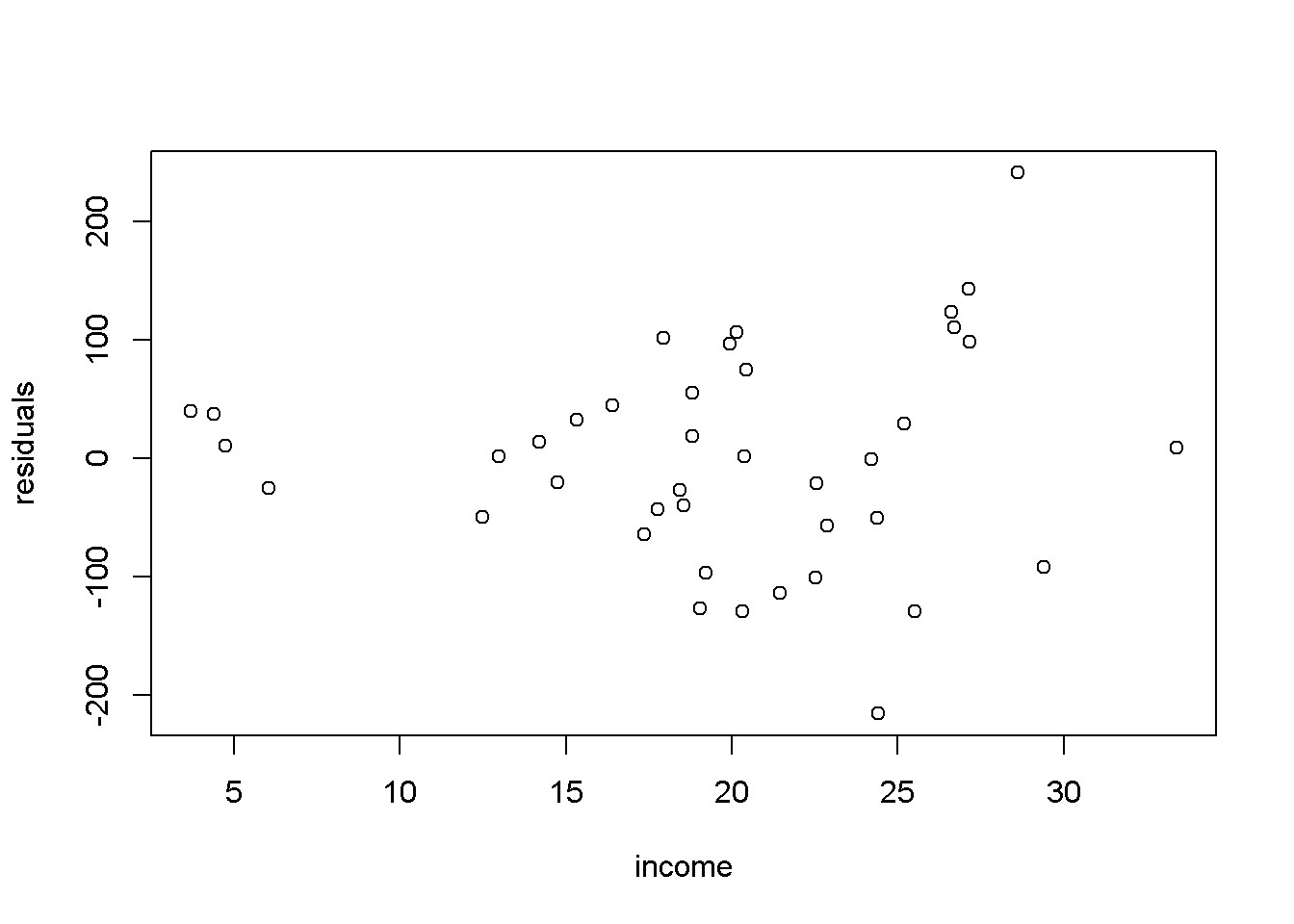


Figure 4.4: Residual plot for the *food* linear-log model 4.5 illustrates this simulated example.

*yi* = 1 + *xi* + *ei, i* = 1*,...,N* (4.11)

**set.seed**(12345) *#sets the seed for the random number generator* x <- **runif**(300, 0, 10) e <- **rnorm**(300, 0, 1) y <- 1+x+e mod3 <- **lm**(y~x) ehat <- **resid**(mod3) **plot**(x,ehat, xlab="x", ylab="residuals")

The next example illustrates how the residuals look like when a linear functional form is used when the true relationship is, in fact, quadratic. The data generating equation is given in Equation 4.12, where *x* is the same uniformly distributed between *−*2*.*5 and 2*.*5), and *e ∼ N*(0*,*4). Figure 4.6 shows the residuals from estimating an incorrectly specified, linear econometric model when the correct specification should be quadratic.

*yi* = 15 *−* 4*x*2*i* + *ei, i* = 1*,...,N* (4.12)

*4.4. RESIDUALS AND DIAGNOSTICS*

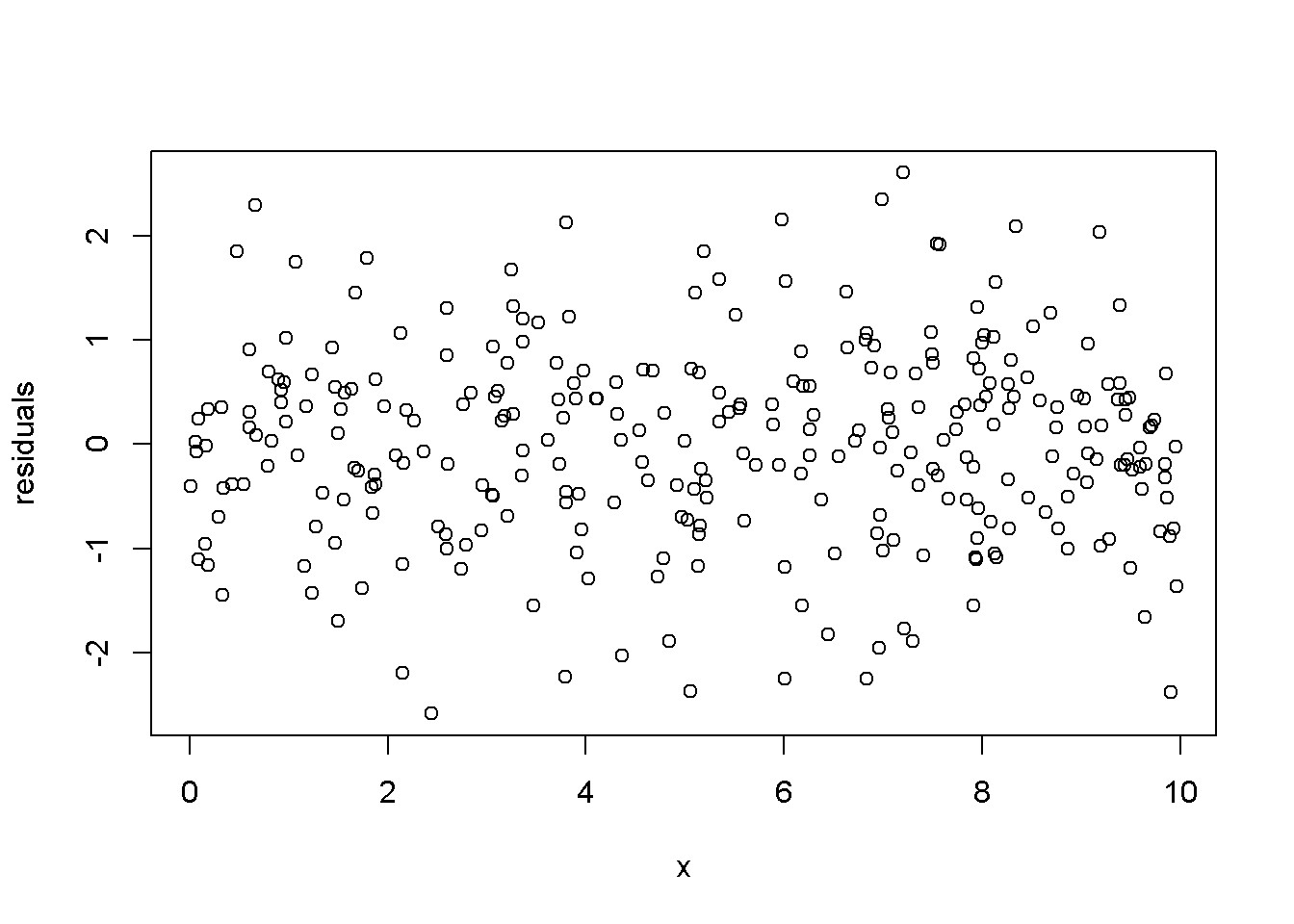


Figure 4.5: Residuals generated by a simulated regression model that satisfies the regression assumptions

|  |
| --- |
| **set.seed**(12345) x <- **runif**(1000, -2.5, 2.5) e <- **rnorm**(1000, 0, 4) y <- 15-4\*x^2+e mod3 <- **lm**(y~x) ehat <- **resid**(mod3) ymi <- **min**(ehat) yma <- **max**(ehat)  **plot**(x, ehat, ylim=**c**(ymi, yma), xlab="x", ylab="residuals",col="grey") |

Another assumption that we would like to test is the normality of the residuals, which assures reliable hypothesis testing and confidence intervals even in small samples. This assumption can be assessed by inspecting a histogram of the residuals, as well as performing a Jarque-Bera test, for which the null hypothesis is “Series is normally distributed”. Thus, a small *p*-value rejects the null hypothesis, which means the series fails the normality test. The Jarque-Bera test requires installing and loading the package tseries in R. Figure 4.7 shows a histogram and a superimposed normal distribution for the linear *food expenditure* model.

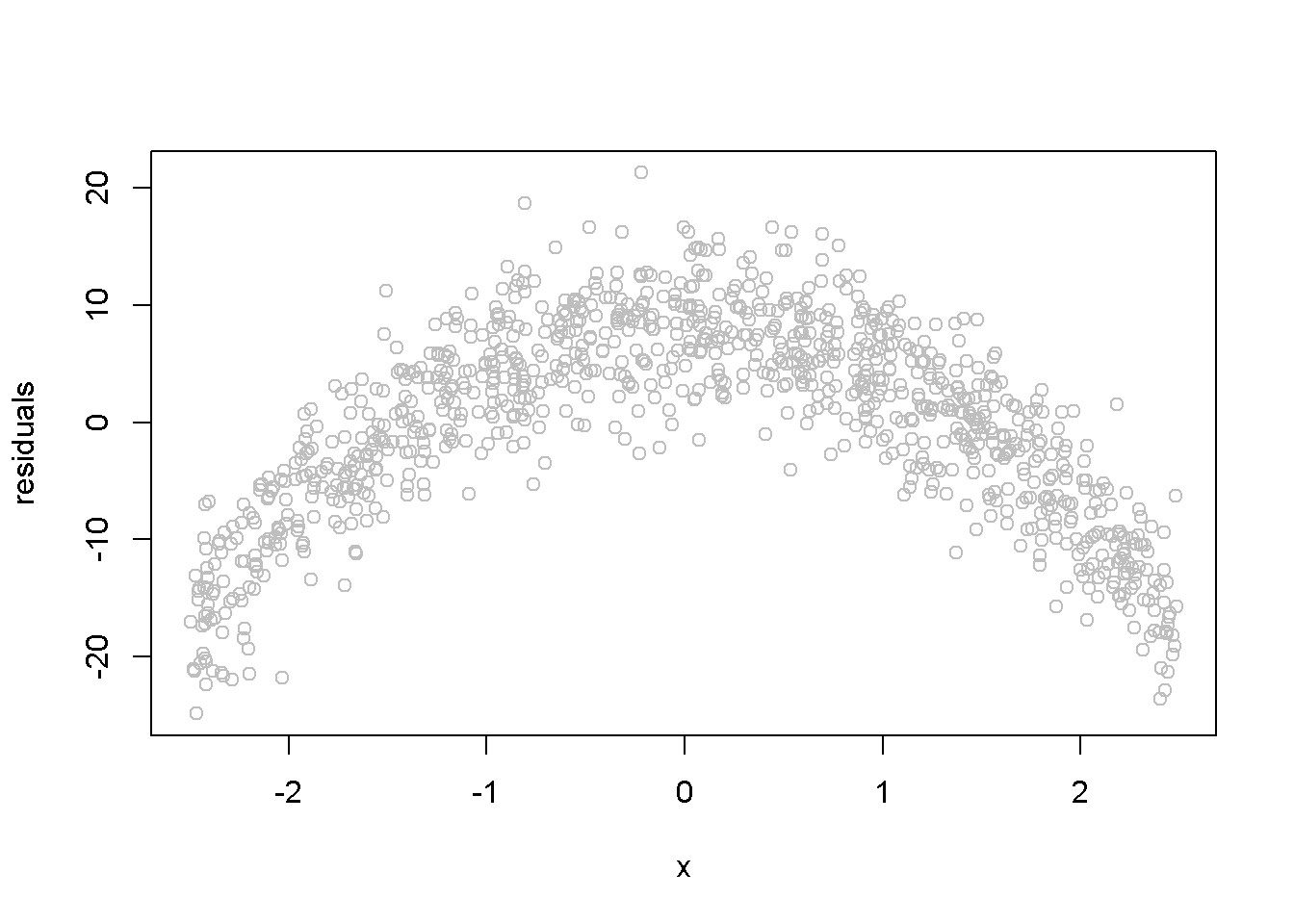


Figure 4.6: Simulated quadratic residuals from an incorrectly specified econometric model

|  |
| --- |
| **library**(tseries)  mod1 <- **lm**(food\_exp~income, data=food) ehat <- **resid**(m1) ebar <- **mean**(ehat) sde <- **sd**(ehat)  **hist**(ehat, col="grey", freq=FALSE, main="", ylab="density", xlab="ehat")  **curve**(**dnorm**(x, ebar, sde), col=2, add=TRUE, ylab="density", xlab="ehat") |

**jarque.bera.test**(ehat) *#(in package 'tseries')*

##

## Jarque Bera Test

##

## data: ehat

## X-squared = 0.06334, df = 2, p-value = 0.969

While the histogram in Figure 4.7 may not strongly support one conclusion or another about the normlity of ehat, the Jarque-Bera test is unambiguous: there is no evidence against the normality hypothesis.

*4.5. POLYNOMIAL MODELS*

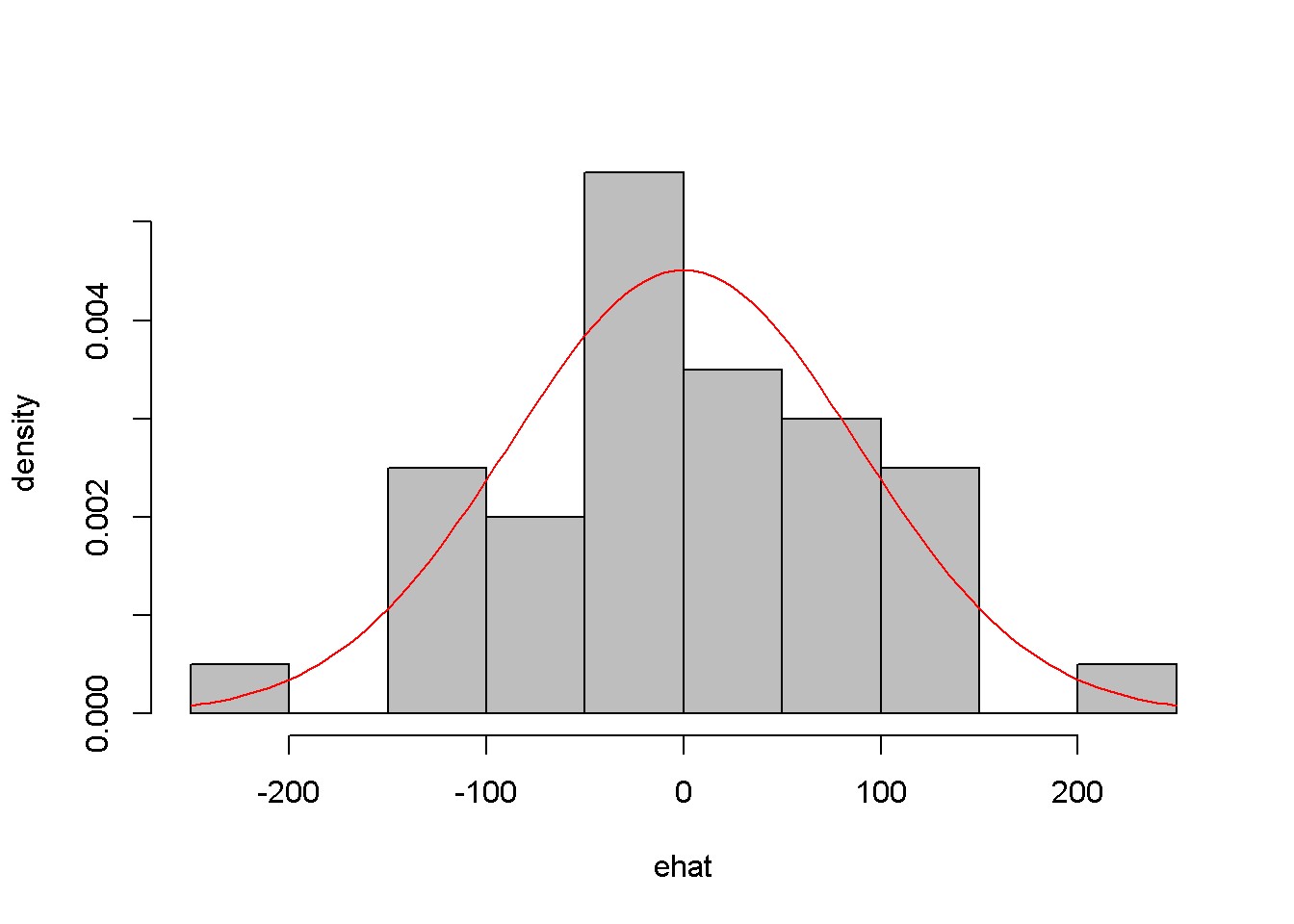


Figure 4.7: Histogram of residuals from the *food* linear model

Lab 5- Polynomial Models

Regression models may include quadratic or cubic terms to better describe the nature of the dadta. The following is an example of quadratic and cubic model for the wa\_wheat dataset, which gives annual wheat yield in tonnes per hectare in

Greenough Shire in Western Australia over a period of 48 years. The *linear* model is given in Equation 4.13, where the subscript *t* indicates the observation period.

*yieldt* = *β*1 + *β*2*timet* + *et* (4.13)

|  |
| --- |
| **library**(PoEdata) **data**("wa\_wheat")  mod1 <- **lm**(greenough~time, data=wa\_wheat) ehat <- **resid**(mod1)  **plot**(wa\_wheat$time, ehat, xlab="time", ylab="residuals") |

Figure 4.8 shows a pattern in the residuals generated by the linear model, which may inspire us to think of a more appropriate functional form, such as the one in Equation 4.14.

*yieldt* = *β*1 + *β*2*time*3*t* + *et* (4.14)

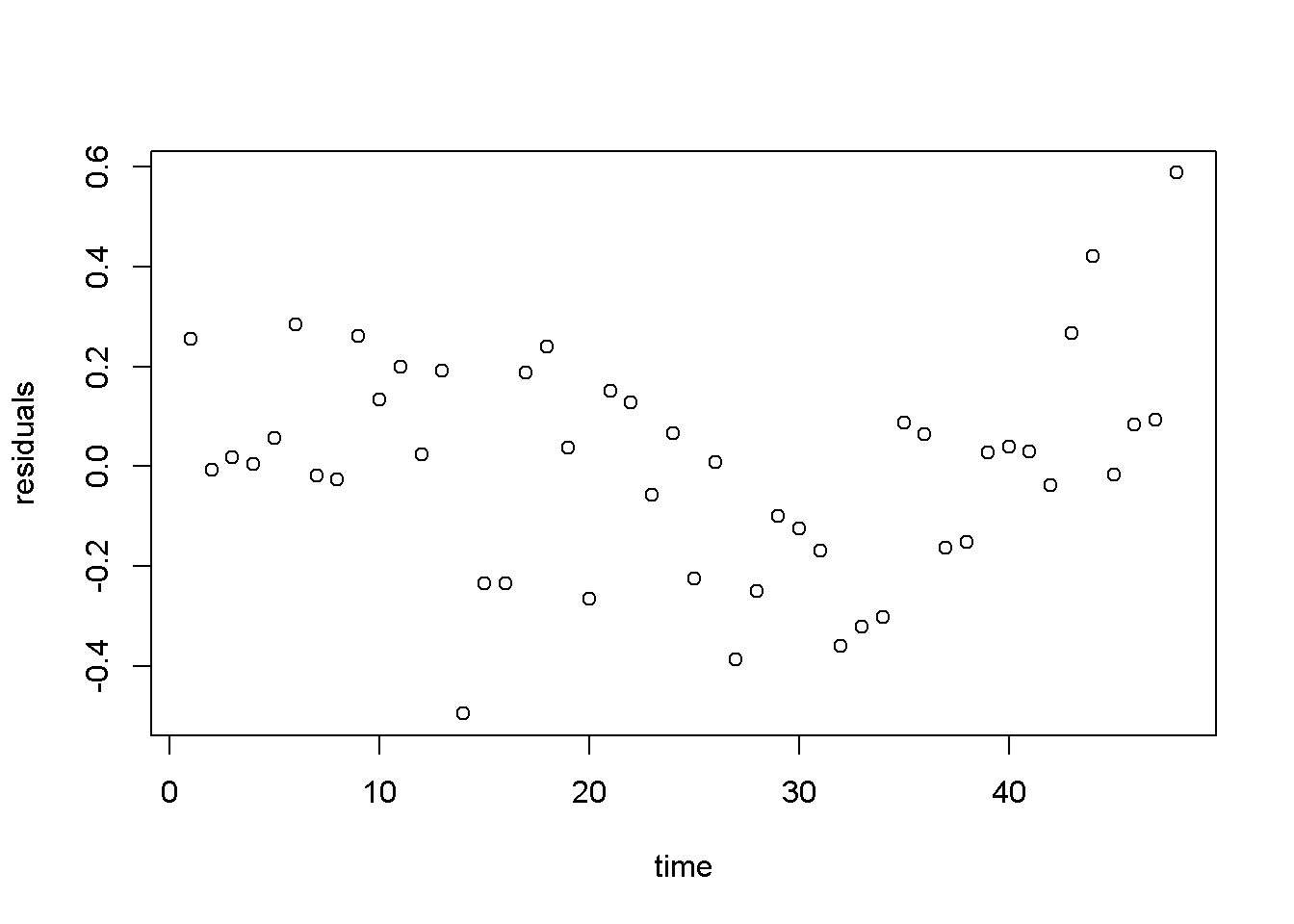


Figure 4.8: Residuals from the linear *wheatyield* model

Please note in the following code sequence the use of the function I(), which is needed in R when an independent variable is transformed by mathematical operators. You do not need the operator I() when an independent variable is transformed through a function such as *log*(*x*). In our example, the transformation requiring the use of I() is raising time to the power of 3. Of course, you can create a new variable, x3=xˆ3 if you wish to avoid the use of I() in a regression equation.

mod2 <- **lm**(wa\_wheat$greenough~**I**(time^3), data=wa\_wheat)

ehat <- **resid**(mod2) **plot**(wa\_wheat$time, ehat, xlab="time", ylab="residuals")

Figure 4.9 displays a much better image of the residuals than Figure 4.8, since the residuals are more evenly spread about the zero line.

## 4.6 Log-Linear Models

Transforming the dependent variable with the *log*() function is useful when the variable has a skewed distribution, which is in general the case with amounts that cannot be negative. The *log*() transformation often makes the distribution closer to normal. The general log-linear model is given in Equation 4.15.

*4.6. LOG-LINEAR MODELS*

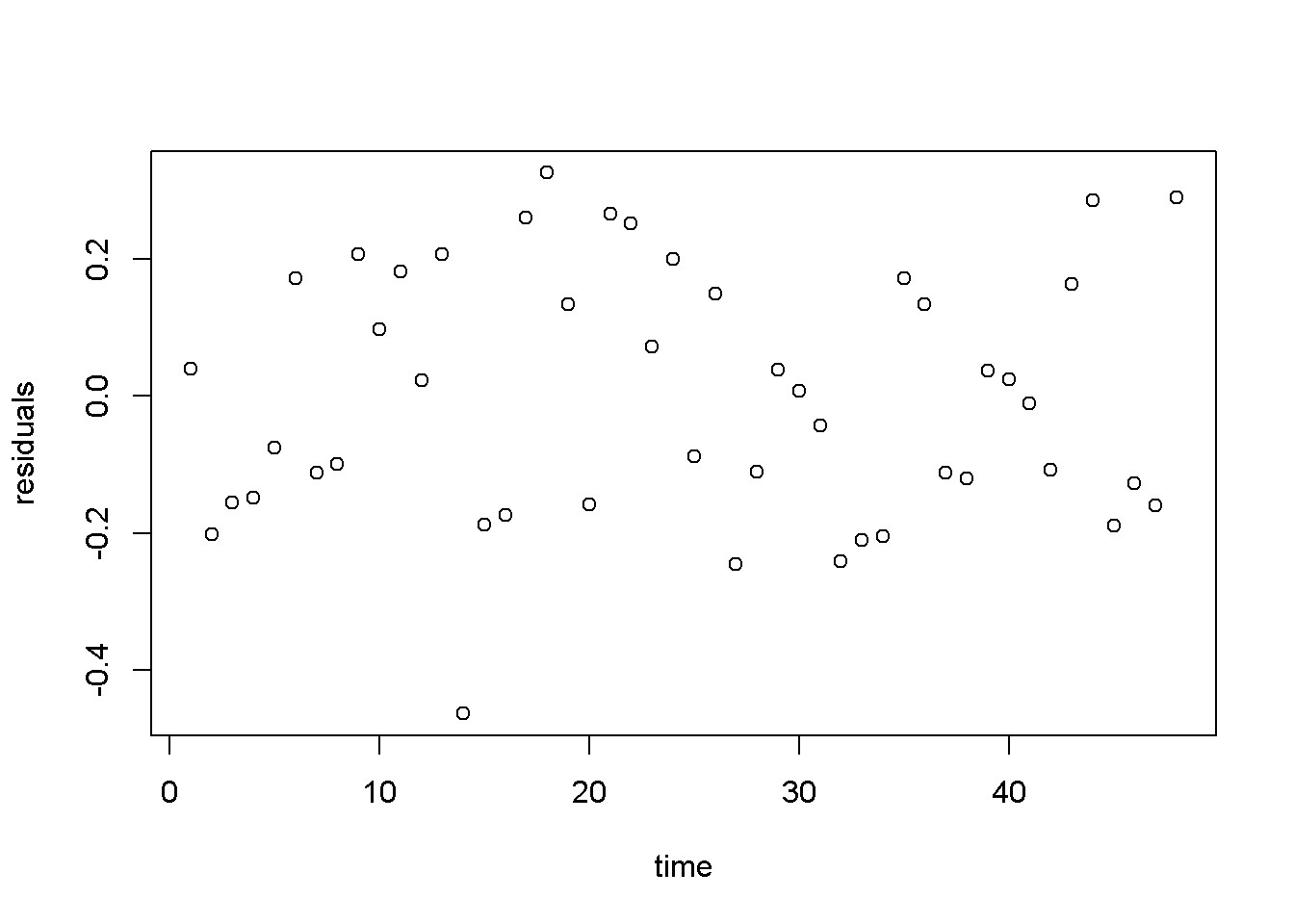


Figure 4.9: Residuals from the cubic *wheatyield* model

*log*(*yi*) = *β*1 + *β*2*xi* + *ei* (4.15)

The following formulas are easily derived from the log-linear Equation 4.15. The semi-elasticity has here a different interpretation than the one in the linear-log model: here, an increase in *x* by one unit (of *x*) produces a change of 100*b*2 percent in *y*. For small changes in *x*, the amount 100*b*2 in the log-linear model can also be interpreted as the growth rate in *y* (corresponding to a unit increase in *x*). For instance, if *x* is time, then 100*b*2 is the growth rate in *y* per unit of time.

* Prediction: *y*ˆ*n* = *exp*(*b*1 + *b*2*x*), or *y*ˆ*c* = *exp*(*b*1 + *b*2*x* + *σ*ˆ22), with the “natural” predictor *y*ˆ*n* to be used in small samples and the “corrected” predictor, *y*ˆ*c*, in large samples
* Marginal effect (slope): *dxdy* = *b*2*y*
* Semi-elasticity: %∆*y* = 100*b*2∆*x*

Let us do these calculations first for the *yield* equation using the *wa\_wheat* dataset.

|  |
| --- |
| mod4 <- **lm**(**log**(greenough)~time, data=wa\_wheat) smod4 <- **summary**(mod4) tbl <- **data.frame**(**xtable**(smod4))  **kable**(tbl, caption="Log-linear model for the \*yield\* equation") |

Table 4.3 gives *b*2 = 0*.*017844, which indicates that the rate of growth in wheat Table 4.3: Log-linear model for the \*yield\* equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | -0.343366 | 0.058404 | -5.87914 | 0 |
| time | 0.017844 | 0.002075 | 8.59911 | 0 |

Table 4.4: Log-linear ’wage’ regression output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 1.609444 | 0.086423 | 18.6229 | 0 |
| educ | 0.090408 | 0.006146 | 14.7110 | 0 |

production has increased at an average rate of approximately 1*.*78 percent per year.

The *wage* log-linear equation provides another example of calculating a growth rate, but this time the independent variable is not *time*, but *education*. The predictions and the slope are calculated for *educ* = 12 years.

|  |
| --- |
| **data**("cps4\_small", package="PoEdata")  xeduc <- 12  mod5 <- **lm**(**log**(wage)~educ, data=cps4\_small) smod5 <- **summary**(mod5) tabl <- **data.frame**(**xtable**(smod5))  **kable**(tabl, caption="Log-linear 'wage' regression output") |
| b1 <- **coef**(smod5)[[1]] b2 <- **coef**(smod5)[[2]] sighat2 <- smod5$sigma^2  g <- 100\*b2 *#growth rate*  yhatn <- **exp**(b1+b2\*xeduc) *#"natural" predictiction* yhatc <- **exp**(b1+b2\*xeduc+sighat2/2) *#corrected prediction*  DyDx <- b2\*yhatn *#marginal effect* |

Here are the results of these calculations: “natural” prediction *y*ˆ*n* = 14*.*796; corrected prediction, *y*ˆ*c* = 16*.*996; growth rate *g* = 9*.*041; and marginal effect *dxdy* = 1*.*34. The growth rate indicates that an increase in education by one unit (see the data description using ?cps4\_small) increases hourly wage by 9*.*041 percent.

Figure 4.10 presents the “natural” and the “corrected” regression lines for the *wage* equation, together with the actual data points.

education=**seq**(0,22,2) yn <- **exp**(b1+b2\*education) yc <- **exp**(b1+b2\*education+sighat2/2)

*4.6. LOG-LINEAR MODELS*

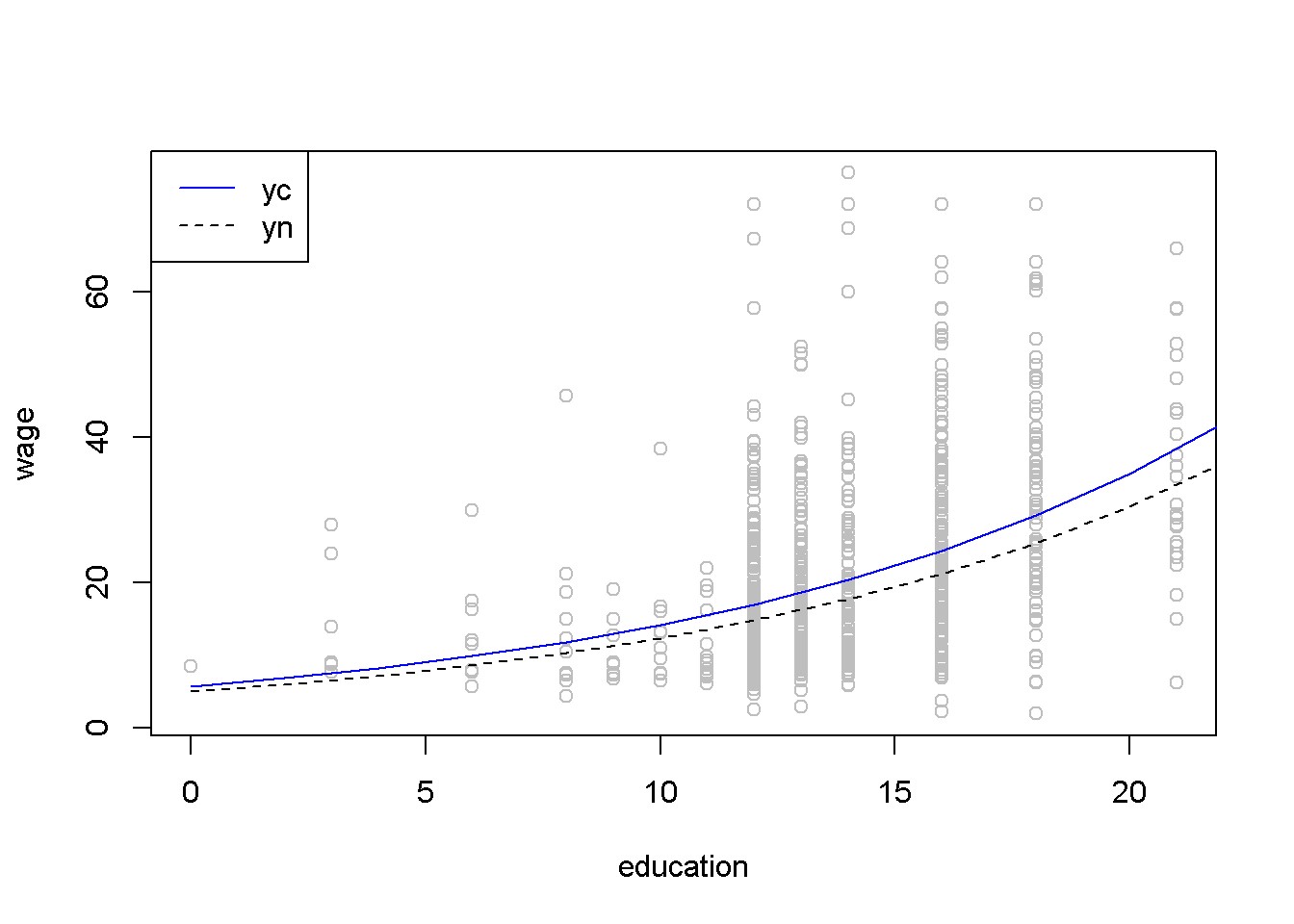


Figure 4.10: The ’normal’ and ’corrected’ regression lines in the log-linear *wage* equation

**plot**(cps4\_small$educ, cps4\_small$wage, xlab="education", ylab="wage", col="grey") **lines**(yn~education, lty=2, col="black") **lines**(yc~education, lty=1, col="blue") **legend**("topleft", legend=**c**("yc","yn"), lty=**c**(1,2), col=**c**("blue","black"))

The regular *R*2 cannot be used to compare two regression models having different dependent variables such as a linear-log and a log-linear models; when such a comparison is needed, one can use the general *R*2, which is *Rg*2 = [*corr*(*y,y*ˆ]2. Let us calculate the generalized *R*2 for the quadratic and the log-linear *wage* models.

|  |
| --- |
| mod4 <- **lm**(wage~**I**(educ^2), data=cps4\_small) yhat4 <- **predict**(mod4)  mod5 <- **lm**(**log**(wage)~educ, data=cps4\_small) smod5 <- **summary**(mod5) b1 <- **coef**(smod5)[[1]] b2 <- **coef**(smod5)[[2]] sighat2 <- smod5$sigma^2  yhat5 <- **exp**(b1+b2\*cps4\_small$educ+sighat2/2) rg4 <- **cor**(cps4\_small$wage, yhat4)^2 |

rg5 <- **cor**(cps4\_small$wage,yhat5)^2

The quadratic model yields *Rg*2 = 0*.*188, and the log-linear model yields *Rg*2 = 0*.*186; since the former is higher, we conclude that the quadratic model is a better fit to the data than the log-linear one. (However, other tests of how the two models meet the assumptions of linear refgression may reach a different conclusion; *R*2 is only one of the model selection criteria.)

To determne a forecast interval estimate in the log-linear model, we first construct the interval in logs using the natural predictor *y*ˆ*n*, then take antilogs of the interval limits. The forecasting error is the same as before, given in Equation 4.4. The following calculations use an education level equal to 12 and *α* = 0*.*05.

|  |
| --- |
| *# The \*wage\* log-linear model # Prediction interval for educ = 12* alpha <- 0.05 xeduc <- 12  xedbar <- **mean**(cps4\_small$educ)  mod5 <- **lm**(**log**(wage)~educ, data=cps4\_small) b1 <- **coef**(mod5)[[1]] b2 <- **coef**(mod5)[[2]] df5 <- mod5$df.residual N <- **nobs**(mod5) tcr <- **qt**(1-alpha/2, df=df5) smod5 <- **summary**(mod5) varb2 <- **vcov**(mod5)[2,2] sighat2 <- smod5$sigma^2  varf <- sighat2+sighat2/N+(xeduc-xedbar)^2\*varb2 sef <- **sqrt**(varf) lnyhat <- b1+b2\*xeduc lowb <- **exp**(lnyhat-tcr\*sef) upb <- **exp**(lnyhat+tcr\*sef) |

The result is the confidence interval (5*.*26*,* 41*.*62). Figure 4.11 shows a 95% confidence band for the log-linear *wage* model.

|  |
| --- |
| *# Drawing a confidence band for the log-linear*  *# \*wage\* equation*  xmin <- **min**(cps4\_small$educ) xmax <- **max**(cps4\_small$educ)+2 education <- **seq**(xmin, xmax, 2) lnyhat <- b1+b2\*education yhat <- **exp**(lnyhat) |

*4.7. THE LOG-LOG MODEL*

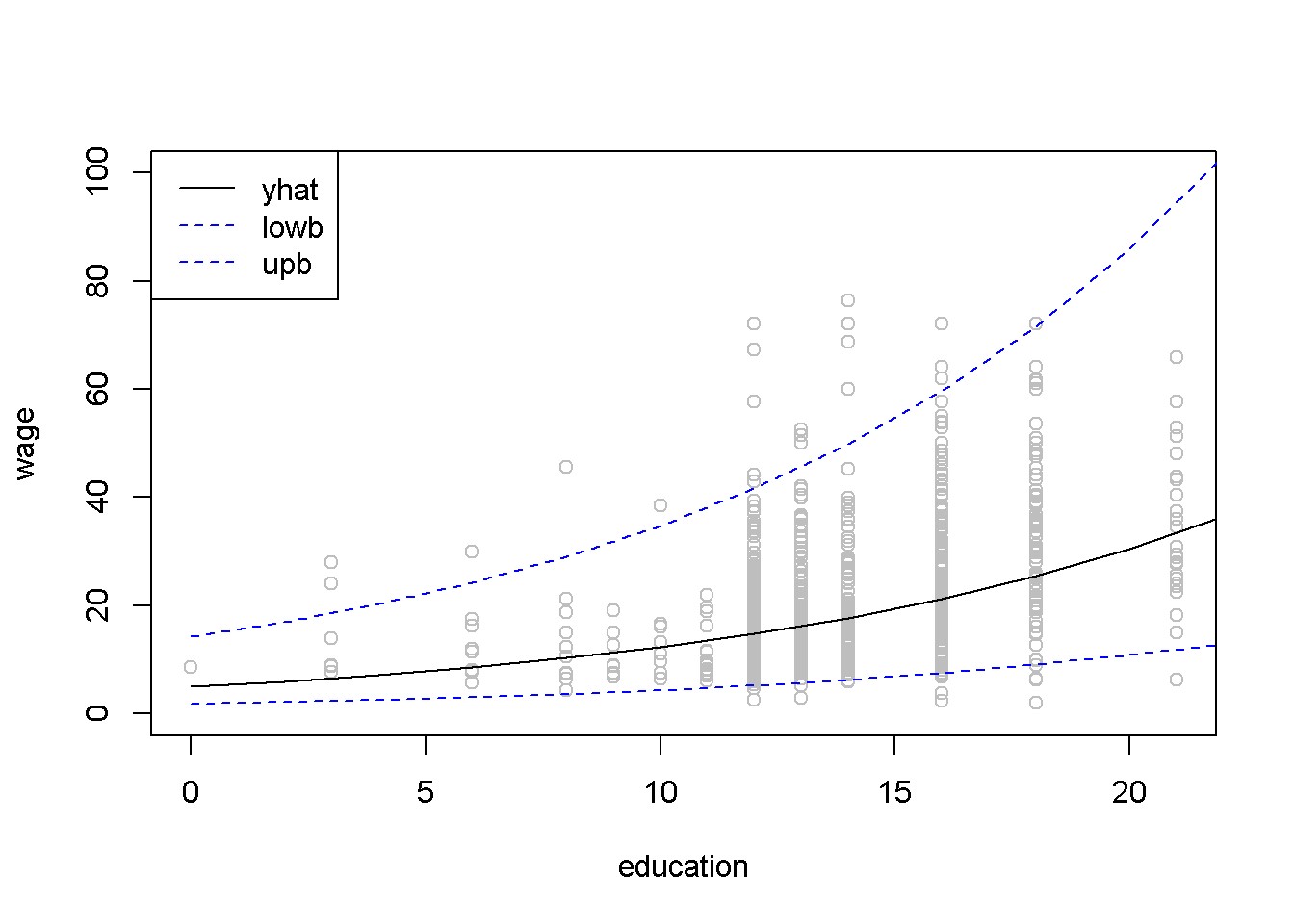


Figure 4.11: Confidence band for the log-linear *wage* equation

|  |
| --- |
| varf <- sighat2+sighat2/N+(education-xedbar)^2\*varb2 sef <- **sqrt**(varf) lowb <- **exp**(lnyhat-tcr\*sef) upb <- **exp**(lnyhat+tcr\*sef)  **plot**(cps4\_small$educ, cps4\_small$wage, col="grey", xlab="education", ylab="wage", ylim=**c**(0,100))  **lines**(yhat~education, lty=1, col="black") **lines**(lowb~education, lty=2, col="blue") **lines**(upb~education, lty=2, col="blue") **legend**("topleft", legend=**c**("yhat", "lowb", "upb"), lty=**c**(1, 2, 2), col=**c**("black", "blue", "blue")) |

## 4.7 The Log-Log Model

The log-log model has the desirable property that the coefficient of the independent variable is equal to the (constant) elasticity of *y* with respect to *x*. Therefore, this model is often used to estimate supply and demand equations. Its standard form is given in Equation 4.16, where *y*, *x*, and *e* are *N ×* 1 vectors.

*log*(*y*) = *β*1 + *β*2*log*(*x*) + *e* (4.16)

Table 4.5: The log-log poultry regression equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 3.71694 | 0.022359 | 166.2362 | 0 |
| log(p) | -1.12136 | 0.048756 | -22.9992 | 0 |
| *# Calculating log-log demand for chicken* **data**("newbroiler", package="PoEdata") mod6 <- **lm**(**log**(q)~**log**(p), data=newbroiler) b1 <- **coef**(mod6)[[1]] b2 <- **coef**(mod6)[[2]] smod6 <- **summary**(mod6) tbl <- **data.frame**(**xtable**(smod6))  **kable**(tbl, caption="The log-log poultry regression equation") | | | | | | |

Table 4.5 gives the log-log regression output. The coefficient on *p* indicates that an increase in price by 1% changes the quantity demanded by *−*1*.*121%.

|  |
| --- |
| *# Drawing the fitted values of the log-log equation* ngrid <- 20 *# number of drawing points* xmin <- **min**(newbroiler$p) xmax <- **max**(newbroiler$p)  step <- (xmax-xmin)/ngrid *# grid dimension* xp <- **seq**(xmin, xmax, step) sighat2 <- smod6$sigma^2  yhatc <- **exp**(b1+b2\***log**(newbroiler$p)+sighat2/2) yc <- **exp**(b1+b2\***log**(xp)+sighat2/2) *#corrected q* **plot**(newbroiler$p, newbroiler$q, ylim=**c**(10,60), xlab="price", ylab="quantity")  **lines**(yc~xp, lty=1, col="black") |

*# The generalized R-squared:* rgsq <- **cor**(newbroiler$q, yhatc)^2

The generalized *R*2, wich uses the corrected fitted values, is equal to 0*.*8818.

*4.7. THE LOG-LOG MODEL*

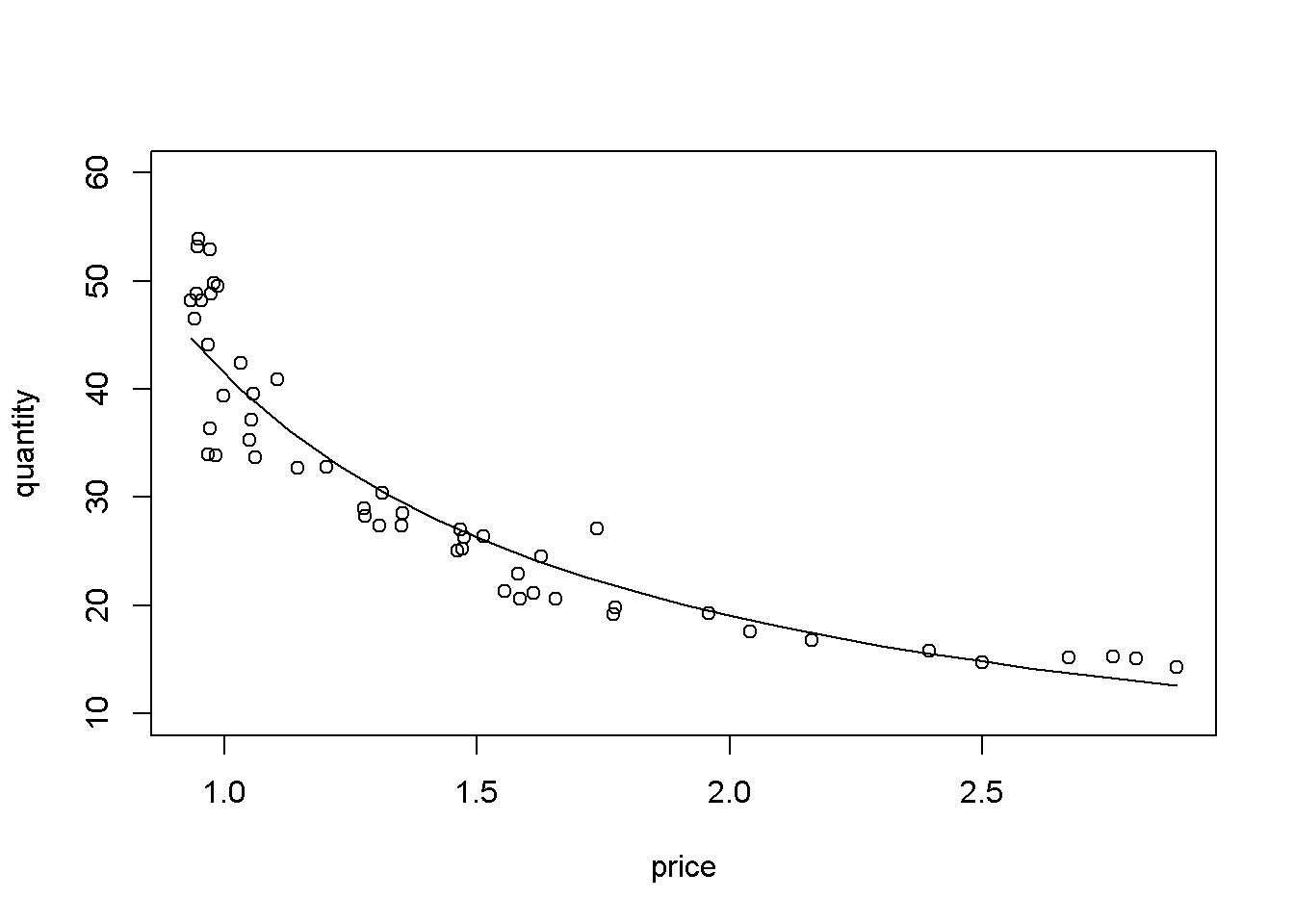


Figure 4.12: Log-log demand for chicken

Lab 6 – Hands-on on your assignment

# 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)