

**Faculty of Business and Law**

**School of Accounting, Finance and Economics**

**BECS2002 / Econometrics and Data Analytics**

**Lab Handbook**

**Week 16**

**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

Lecture 1: The Multiple Regression Model, Example and Interval Estimation 4

Lecture 2 - Hypothesis Testing in Multiple Regression 12

Video 1 - Interaction Terms in Linear Regression 18

**Video 2 -** Further Inference in Multiple Regression 22

Lab 1 - Omitted Variable Bias 28

Lab 2 - Collinearity 34

Lab 3 - Comparing Two Regressions: the Chow Test 41

Lab 4 -The Difference-in-Differences Estimator 49

Lab 5 -- Using Panel Data 53

Lab 6 - Heteroskedasticity 59

Lecture 1: The Multiple Regression Model, Example and Interval Estimation

**rm**(list=**ls**())

**library**(PoEdata)

**library**(knitr)

**library**(xtable)

**library**(printr)

**library**(effects)

**library**(car)

**library**(AER)

**library**(broom)

This chapter uses a few new packages: effects (Fox et al. 2016), car (Fox and Weisberg 2016), and AER, (Kleiber and Zeileis 2015).

## 5.1 The General Model

A multiple regression model is very similar to the simple regression model, but includes more independent variables. Thus, the interpretation of a slope parameter has to take into account possible changes in other independent variables: a slope parameter, say *βk*, gives the change in the dependent variable, *y*, when the independent variable *xk* increases by one unit **while all the other independent variables remain constant**. Equation 5.1 gives the general form of a multiple regression model, where *y*, *xk*, and *e* are *N ×* 1 vectors, *N* is the number of observations in the sample, and *k* = 1*,...,K* indicates the *k*-th independent variable. The notation used in Equation 5.1 implies that the first independent variable, *x*1, is an *N ×* 1 vector of 1s.

*y* = *β*1 + *β*2*x*2 + *...* + *βKxK* + *e* (5.1)

Table 5.1: Summary statistics for dataset *andy*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| column | n | mean | sd | median | min | max |
| sales | 75 | 77.3747 | 6.488537 | 76.50 | 62.40 | 91.20 |
| price | 75 | 5.6872 | 0.518432 | 5.69 | 4.83 | 6.49 |
| advert | 75 | 1.8440 | 0.831677 | 1.80 | 0.50 | 3.10 |

The model assumptions remain the same, with the additional requirement that no independent variable is a linear combination of the others.

## 5.2 Example: Big Andy’s Hamburger Sales

The *andy* dataset includes variables *sales*, which is monthly revenue to the company in $1000s, *price*, which is a price index of all products sold by Big Andy’s, and *advert*, the advertising expenditure in a given month, in $1000s. Summary statistics for the *andy* dataset is shown in Table 5.1. The basic *andy* model is presented in Equation 5.2.

*sales* = *β*1 + *β*2*price* + *β*3*advert* + *e* (5.2)

|  |
| --- |
| *# Summary statistics*  **data**(andy)  s=**tidy**(andy)[,**c**(1:5,8,9)]  **kable**(s,caption="Summary statistics for dataset $andy$") |
| *# The basic \*andy\* model* **data**("andy",package="PoEdata")  mod1 <- **lm**(sales~price+advert, data=andy)  smod1 <- **data.frame**(**xtable**(**summary**(mod1)))  **kable**(smod1, caption="The basic multiple regression model", col.names= **c**("coefficient", "Std. Error", "t-value", "p-value"), align="c", digits=3) |

Table 5.2 shows that, for any given (but fixed) value of advertising expenditure, an increase in price by $1 decreases sales by $7908. On the other hand, for any given price, sales increase by $1863 when advertising expenditures increase by $1000.

effprice <- **effect**("price", mod1)

**plot**(effprice)

**summary**(effprice)

Table 5.2: The basic multiple regression model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | coefficient | Std. Error | t-value | p-value |
| (Intercept) | 118.914 | 6.352 | 18.722 | 0.000 |
| price | -7.908 | 1.096 | -7.215 | 0.000 |
| advert | 1.863 | 0.683 | 2.726 | 0.008 |

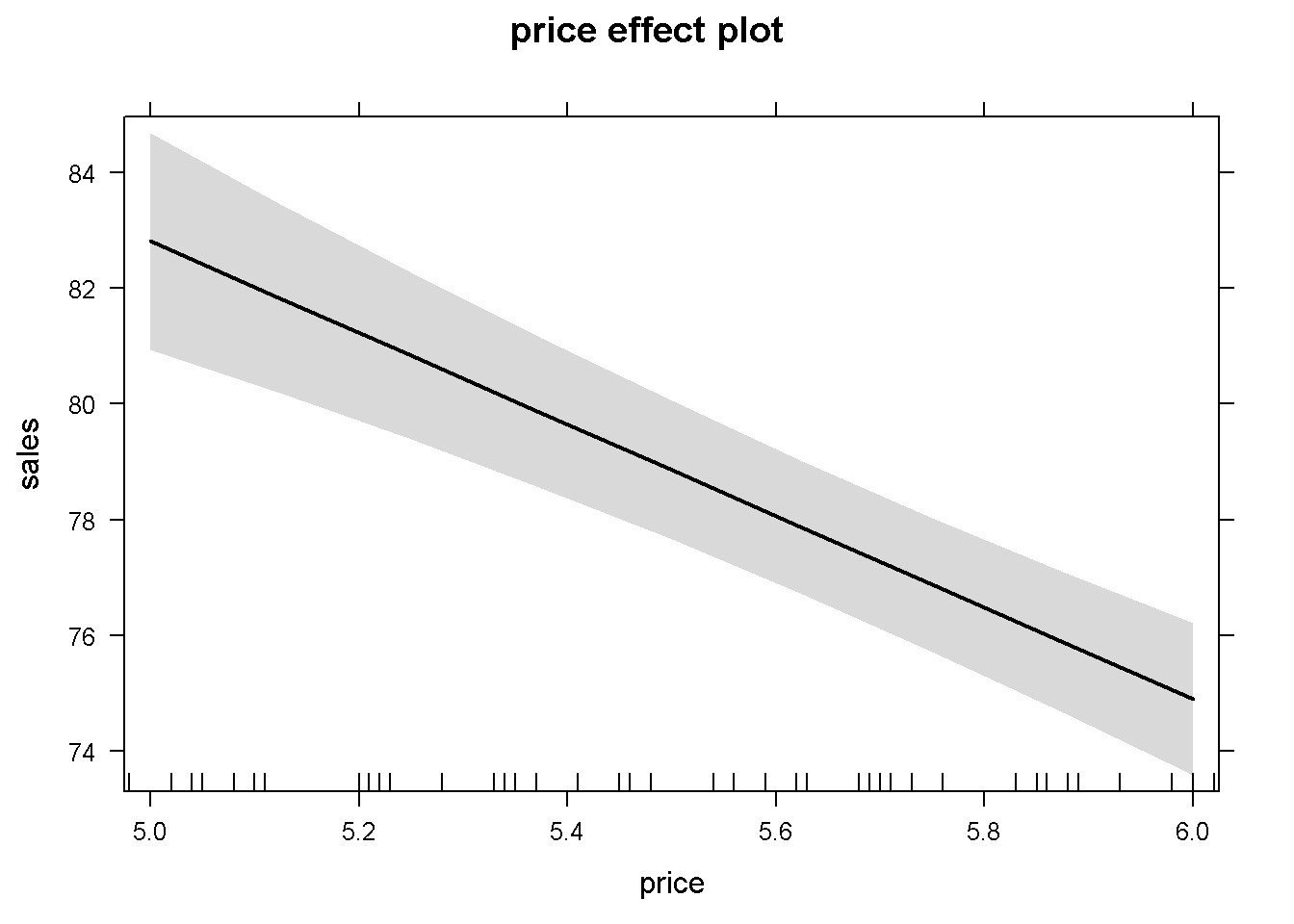


Figure 5.1: The partial effect of *price* in the basic *andy* regression

##

## price effect

## price

## 5 5.5 6 ## 82.8089 78.8550 74.9011

##

## Lower 95 Percent Confidence Limits

## price

## 5 5.5 6 ## 80.9330 77.6582 73.5850

##

## Upper 95 Percent Confidence Limits

## price

## 5 5.5 6

## 84.6849 80.0518 76.2172

Figure 5.1 shows the predicted levels of the dependent variable *sales* and its 95% confidence band for the sample values of the variable *price*. In more complex functional forms, the *R* function effect() plots the partial effect of a variable for given levels of the other independent variables in the model. The simplest possible call of this function requires, as arguments, the name of the term for which we wish the partial effect (in our example *price*), and the object (model) in question. If not otherwise specified, the confidence intervals (band) are determined for a 95% confidence level and the other variables at their means. A simple use of the effects package is presented in Figure 5.2, which plots the partial effects of all variables in the basic *andy* model. (Function effect() plots only one graph for one variable, while allEffects() plots all variables in the model.)

alleffandy <- **allEffects**(mod1)

**plot**(alleffandy)

*# Another example of using the function effect()*

mod2 <- **lm**(sales~price+advert+**I**(advert^2), data=andy)

**summary**(mod2)

##

## Call:

## lm(formula = sales ~ price + advert + I(advert^2), data = andy)

##

## Residuals:

## Min 1Q Median 3Q Max

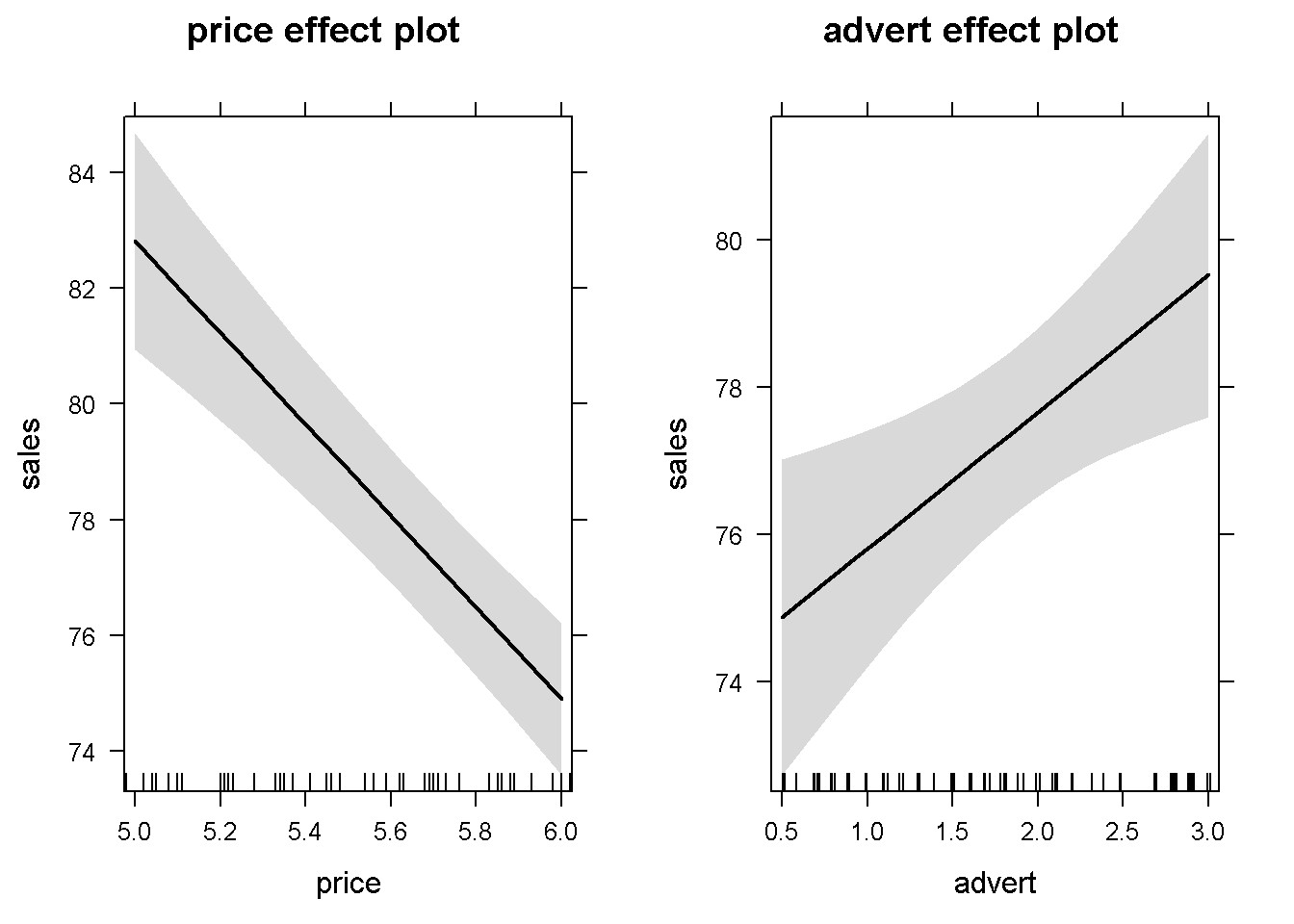


Figure 5.2: Using the function ’allEffects()’ in the basic *andy* model

## -12.255 -3.143 -0.012 2.851 11.805

##

## Coefficients:

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

## (Intercept) 109.719 6.799 16.14 < 2e-16 \*\*\* ## price -7.640 1.046 -7.30 3.2e-10 \*\*\*

## advert 12.151 3.556 3.42 0.0011 \*\* ## I(advert^2) -2.768 0.941 -2.94 0.0044 \*\*

## ---

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

##

## Residual standard error: 4.65 on 71 degrees of freedom

## Multiple R-squared: 0.508, Adjusted R-squared: 0.487 ## F-statistic: 24.5 on 3 and 71 DF, p-value: 5.6e-11

**plot**(**effect**("I(advert^2)", mod2))

Figure 5.3 is an example of using the effect() function to plot the partial effect of a quadratic independent variable. I chose to insert the I(advertˆ2) term to indicate that the variable of interest needs to be specified exactly as it appears in the model.

All the methods available in *R* for simple linear regression models are available for multiple models as well. Thus, to extract the information generated by the lm() function, we use the similar code as before. As we have already learned, the command ?mod1 gives a list with all the information stored in the mod1 object. The next code sequence illustrates how to access various regression output items for the basic *andy* equation.

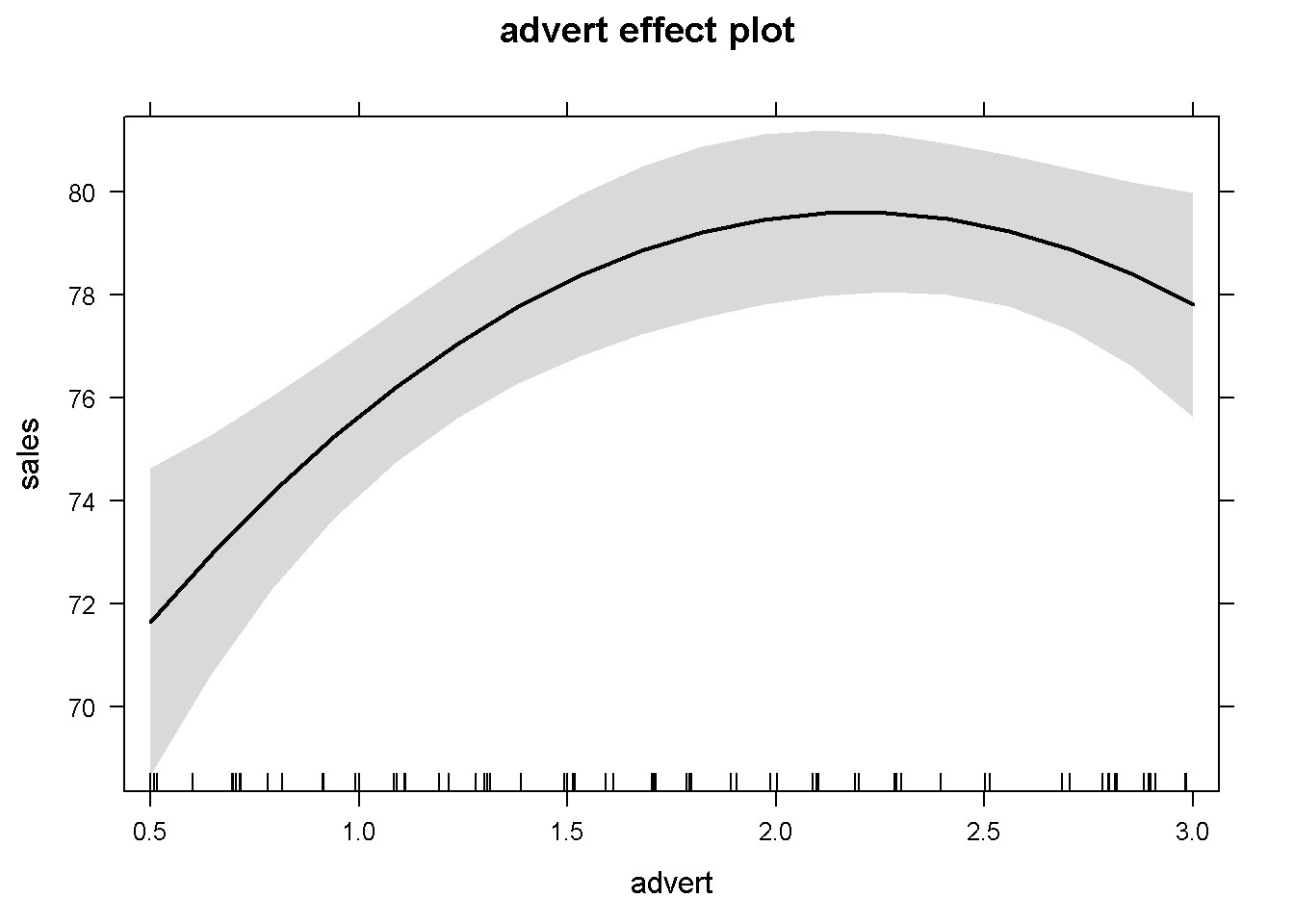


Figure 5.3: An example of using the function ’effect’ in a quadratic model

|  |
| --- |
| mod1 <- **lm**(sales~price+advert, data=andy)  smod1 <- **summary**(mod1)  df <- mod1$df.residual  N <- **nobs**(mod1)  b1 <- **coef**(mod1)[[1]] *#or*  b2 <- **coef**(mod1)[["price"]]  b3 <- **coef**(mod1)[["advert"]]  sighat2 <- smod1$sigma^2  anov <- **anova**(mod1)  SSE <- anov[3,2]  SST <- **sum**(anov[,2]) *#sum of column 2 in anova*  SSR <- SST-SSE  **kable**(**data.frame**(**vcov**(mod1)), align='c', digits=3, caption="The coefficient covariance matrix", col.names=**c**("(Intercept)", "price", "advert")) |

The *covariance matrix*, or the *variance-covariance* matrix shown in Table 5.3 contains all the estimated variances and covariances of the regression coefficients. These are useful when testing hypotheses about individual coefficients or combinations of those.

Table 5.3: The coefficient covariance matrix

|  |  |  |  |
| --- | --- | --- | --- |
|  | (Intercept) | price | advert |
| (Intercept) | 40.343 | -6.795 | -0.748 |
| price | -6.795 | 1.201 | -0.020 |
| advert | -0.748 | -0.020 | 0.467 |

## 5.3 Interval Estimation in Multiple Regression

Interval estimation is similar with the one we have studied in the simple regression model, except the number of degrees of freedom is now *N − K*, where *K* is the number of regression coefficients to be estimated. In the *andy* example, *df* = 72 and the variances and covariances of the estimates are given in the following code sequence.

|  |
| --- |
| varb1 <- **vcov**(mod1)[1,1]  varb2 <- **vcov**(mod1)[2,2]  varb3 <- **vcov**(mod1)[3,3]  covb1b2 <- **vcov**(mod1)[1,2]  covb1b3 <- **vcov**(mod1)[1,3]  covb2b3 <- **vcov**(mod1)[2,3]  seb2 <- **sqrt**(varb2) *#standard error of b2*  seb3 <- **sqrt**(varb3) |

With the calculated standard error of *b*2, *se*(*b*2) = 1*.*096, we can now determine a 95% confidence interval, as shown in the following code lines. The code also shows using the *R* function confint(model, parm, level) to check our results.

alpha <- 0.05

tcr <- **qt**(1-alpha/2, df)

lowb2 <- b2-tcr\*seb2

upb2 <- b2+tcr\*seb2

lowb3 <- b3-tcr\*seb3

upb3 <- b3+tcr\*seb3

confints <- **confint**(mod1, parm=**c**("price", "advert"), level=0.95)

**kable**(**data.frame**(confints),caption="Confidence intervals for 'price' and 'advert'", align="c", col.names=**c**("lowb", "upb"), digits=4)

Finding an interval estimate for a linear combination of the parameters is often needed. Suppose one is interested in determining an interval estimate for *sales* when the price decreases by 40 cents and the advertising expenditure increases by $800 (see Equation 5.3) in the *andy* basic equation.

Table 5.4: Confidence intervals for ’price’ and ’advert’

|  |  |  |
| --- | --- | --- |
|  | lowb | upb |
| price | -10.0927 | -5.7230 |
| advert | 0.5007 | 3.2245 |

*λ* = 0*β*1 *−* 0*.*4*β*2 + 0*.*8*β*3 (5.3)

|  |
| --- |
| alpha <- 0.1  tcr <- **qt**(1-alpha/2, df)  a1 <- 0  a2 <- -0.4  a3 <- 0.8  L <- a1\*b1+a2\*b2+a3\*b3  varL <- a1^2\*varb1+a2^2\*varb2+a3^2\*varb3+  2\*a1\*a2\*covb1b2+2\*a1\*a3\*covb1b3+2\*a2\*a3\*covb2b3  seL <- **sqrt**(varL)  lowbL <- L-tcr\*seL  upbL <- L+tcr\*seL |

The calculated confidence interval is (3*.*471*,*5*.*836). Let us calculate the variance of a linear combination of regression parameters in a more general way to take advantage of *R*’s excellent capabilities of working with complex data structures such as lists, vectors, and matrices. This code sequence introduces a few new elements, such as matrix transposition and multiplication, as well as turning a list into a vector. The matrix multiplication operator in *R* is %\*% and the transposition operator is t().

|  |
| --- |
| a<- **c**(0, -0.4, 0.8) *# vector*  b<- **as.numeric**(**coef**(mod1))*# vector of coefficients*  L <- L <- **sum**(a\*b) *# sum of elementwise products*  V <- **vcov**(mod1) *# the variance-covariance matrix*  A <- **as.vector**(a) *# (indeed not necessary)*  varL <- **as.numeric**(**t**(A) %\*% V %\*% A) |

Lecture 2 - Hypothesis Testing in Multiple Regression

The process of testing hypotheses about a single parameter is similar to the one we’ve seen in simple regression, the only difference consisting in the number of degrees of freedom. As an example, let us test the significance of *β*2 of the basic *andy* equation. The hypotheses are given in Equation 5.4.

*H*0 : *β*2 = 0*, HA* : *β*2 *≠* 0 (5.4)

alpha <- 0.05

df <- mod1$df.residual

tcr <- **qt**(1-alpha/2, df)

b2 <- **coef**(mod1)[["price"]]

seb2 <- **sqrt**(**vcov**(mod1)[2,2])

t <- b2/seb2

The calculated *t* is equal to *−*7*.*215, which is less than *−tcr* = *−*1*.*993, indicating that the null hypothesis is rejected and that *β*2 is significantly different from zero.

As usual, we can perform the same test using the *p*-value instead of *tcr*.

t <- b2/seb2

pval <- 2\*(1-**pt**(**abs**(t), df)) *#two-tail test*

The calculated *p*-value is 4*.*423999 *×* 10*−*10. Since this is less than *α* we reject, again, the null hypothesis *H*0 : *β*2 = 0. Let us do the same for *β*3:

alpha <- 0.05

df <- mod1$df.residual

tcr <- **qt**(1-alpha/2, df)

b3 <- **coef**(mod1)[[3]]

seb3 <- **sqrt**(**vcov**(mod1)[3,3])

tval <- b3/seb3

Calculated *t* = 2*.*726, and *tcr* = 1*.*993. Since *t > tcr* we reject the null hypothesis and conclude that there is a statistically significant relationship between *advert* and *sales*. The same result can be obtained using the *p*-value method:

pval <- 2\*(1-**pt**(**abs**(tval), df))

Result: *p*-value = 0*.*008038 *< α*. *R* shows the two-tail *t* and *p* values for coefficients in regression output (see Table 5.2).

A one-tail hypothesis testing can give us information about price elasitcity of demand in the basic *andy* equation in Table 5.2. Let us test the hypothesis described in Equation 5.5 at a 5% significance level. The calculated value of *t* is the same, but *tcr* corresponds now not to  but to *α*.

*H*0 : *β*2 *≥* 0*, HA* : *β*2 *<* 0 (5.5)

|  |
| --- |
| alpha <- 0.05  tval <- b2/seb2  tcr <- -**qt**(1-alpha, df) *#left-tail test*  pval <- **pt**(tval, df) |

The results *t* = *−*7*.*215, *tcr* = *−*1*.*666 show that the calculated *t* falls in the (left-tail) rejection region. The *p*-value, which is equal to 2*.*211999 *×* 10*−*10, is less than *α*, also rejects the null hypothesis.

Here is how the *p*-values should be calculated, depending on the type of test:

* Two-tail test (*HA* : *β*2 *6*= 0), p-value <- 2\*(1-pt(abs(t), df))
* Left-tail test (*HA* : *β*2 *<* 0), p-value <- pt(t, df)
* Right-tail test (*HA* : *β*2 *>* 0), p-value <- 1-pt(t, df)

Another example of a one-tail hypothesis testing is to test whether an increase of $1 in advertising expenditure increases revenue by more than $1. In the hypothesis testing language, we need to test the hypothesis presented in Equation 5.6.

*H*0 : *β*3 *≤* 1*, HA* : *β*3 *>* 1 (5.6)

tval <- (b3-1)/seb3

pval <- 1-**pt**(tval,df)

The calculated *p*-value is 0*.*105408, which is greater than *α*, showing that we cannot reject the null hypothesis *H*0 : *β*3 *≤* 1 at *α* = 0*.*05. In other words, increasing advertising expenditure by $1 may or may not increase sales by more than $1.

Testing hypotheses for linear combinations of coefficients resembles the interval estimation procedure. Suppose we wish to determine if lowering the price by 20 cents increases sales more than would an increase in advertising expenditure by $500. The hypothesis to test this conjecture is given in Equation 5.7.

*H*0 : 0*β*1 *−* 0*.*2*β*2 *−* 0*.*5*β*3 *≤* 0*, HA* : 0*β*1 *−* 0*.*2*β*2 *−* 0*.*5*β*3 *>* 0 (5.7)

Let us practice the matrix form of testing this hypotesis. *R* functions having names that start with as. coerce a certain structure into another, such as a named list into a vector (names are removed and only numbers remain).

|  |
| --- |
| A <- **as.vector**(**c**(0, -0.2, -0.5))  V <- **vcov**(mod1)  L <- **as.numeric**(**t**(A) %\*% **coef**(mod1))  seL <- **as.numeric**(**sqrt**(**t**(A) %\*% V %\*% A))  tval <- L/seL  pval <- 1-**pt**(tval, df) *# the result (p-value)* |

The answer is *p*-value= 0*.*054619, which barely fails to reject the null hypothesis. Thus, the conjecture that a decrease in price by 20 cents is more effective than an increase in advertising by $800 is not supported by the data.

For two-tail linear hypotheses, *R* has a built-in function, linearHypothesis(model, hypothesis), in the package car. This function tests hypotheses based not on *t*-statistics as we have done so far, but based on an *F*-statistic. However, the *p*-value criterion to reject or not the null hypothesis is the same: reject if *p*-value is less than *α*. Let us use this function to test the two-tail hypothesis similar to the one given in Equation 5.7. (Note that the linearHypothesis() function can test not only one hypothesis, but a set of simultaneous hypotheses.)

hypothesis <- "-0.2\*price = 0.5\*advert"

test <- **linearHypothesis** (mod1, hypothesis)

Fstat <- test$F[2]

pval <- 1-**pf**(Fstat, 1, df)

The calculated *p*-value is 0*.*109238, which shows that the null hypothesis cannot be rejected. There are a few new elements, besides the use of the linearHypothesis function in this code sequence. First, the linearHypothesis() function creates an *R* object that contains several items, one of which is the *F*-statistic we are looking for. This object is shown in Table 5.5 and it is named test in our code. The code element test$F[2] extracts the *F*-statistic from the linearHypothesis() object.

Second, please note that the *F*-statistic has two ‘degrees of freedom’ parameters, not only one as the *t*-statistic does. The first degree of freedom is equal to the number of simultaneous hypotheses to be tested (in our case only one); the second is the number of degrees of freedom in the model, *df* = *N − K*.

Last, the function that calculates the *p*-value is pf(Fval, df1, df2), where Fval is the calculated value of the *F*-statistic, and df1 and df2 are the two degrees of freedom parameters of the *F*-statistic.

**kable**(test, caption="The `linearHypothesis()` object")

Table 5.5: The ‘linearHypothesis()‘ object

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| 73 | 1781.73 | NA | NA | NA | NA |
| 72 | 1718.94 | 1 | 62.7874 | 2.62993 | 0.109238 |

Table 5.6: The quadratic version of the *andy* model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std. Error | t | p-Value |
| (Intercept) | 109.719 | 6.799 | 16.137 | 0.000 |
| price | -7.640 | 1.046 | -7.304 | 0.000 |
| advert | 12.151 | 3.556 | 3.417 | 0.001 |
| I(advert^2) | -2.768 | 0.941 | -2.943 | 0.004 |

## 5.5 Polynomial Regression Models

A polynomial multivariate regression model may include several independent variables at various powers. In such models, the partial (or marginal) effect of a regressor *xk* on the response *y* is determined by the partial derivative *∂x∂yk*. Let us consider again the basic *andy* model with the added *advert* quadratic term as presented in Equation 5.8.

*salesi* = *β*1 + *β*2*pricei* + *β*3*adverti* + *β*4*advert*2*i* + *ei* (5.8)

As we have noticed before, the quadratic term is introduced into the model using the function I() to indicate the presence of a mathematical operator. Table 5.6 presents a summary of the regression output from the model described in Equation 5.8.

|  |
| --- |
| mod2 <- **lm**(sales~price+advert+**I**(advert^2),data=andy)  smod2 <- **summary**(mod2)  tabl <- **data.frame**(**xtable**(smod2))  **names**(tabl) <- **c**("Estimate", "Std. Error", "t", "p-Value")  **kable**(tabl, digits=3, align='c', caption="The quadratic version of the $andy$ model") |

Let us calculate the marginal effect of *advert* on *sales* for two levels of *advert*; the relevant partial derivative is the one in Equation 5.9.

A math equation with black text

Description automatically generated(5.9)

advlevels <- **c**(0.5, 2)

b3 <- **coef**(mod2)[[3]]

b4 <- **coef**(mod2)[[4]]

DsDa <- b3+b4\*advlevels

The calculated marginal effects for the two levels of advertising expenditure are, respectively, 10*.*767254 and 6*.*615309, which shows, as expected, diminishing returns to advertising at any given price.

Often, marginal effects or other quantities are non-linear functions of the parameters of a regression model. The *delta* method allows calculating the variance of such quantities using a Taylor series approximation. This method is, howver, valid only in a vicinity of a point, as any mathematical object involving derivatives.

Suppose we wish to test a hypothesis such as the one in Equation 5.10, where *c* is a constant.

*H*0 : *g*(*β*1*,β*2) = *c* (5.10)

The *delta* method consists in estimating the variance of the function *g*(*β*1*,β*2) around some given data point, using the formula in Equation 5.11, where *gi* stands for calculated at the point *β* = *b*.

*var*(*g*) = *g*12*var*(*b*1) + *g*22*var*(*b*2) + 2*g*1*g*2*cov*(*b*1*,b*2) (5.11)

Let us apply this method to find a confidence interval for the optimal level of advertising in the *andy* quadratic equation, *g*, which is given by Equation 5.12.

A math equation with numbers and symbols

Description automatically generated(5.12)

Equation 5.13 shows the derivatives of function *g*.

A math equations with numbers

Description automatically generated with medium confidence(5.13)

|  |
| --- |
| alpha <- 0.05  df <- mod2$df.residual  tcr <- **qt**(1-alpha/2, df)  g <- (1-b3)/(2\*b4)  g3 <- -1/(2\*b4)  g4 <- -(1-b3)/(2\*b4^2)  varb3 <- **vcov**(mod2)[3,3]  varb4 <- **vcov**(mod2)[4,4]  covb3b4 <- **vcov**(mod2)[3,4]  varg <- g3^2\*varb3+g4^2\*varb4+2\*g3\*g4\*covb3b4  seg <- **sqrt**(varg)  lowbg <- g-tcr\*seg  upbg <- g+tcr\*seg |

The point estimate of the optimal advertising level is 2*.*014, with its confidence interval (1*.*758, 2*.*271).

Video 1 - Interaction Terms in Linear Regression

Interaction terms in a regression model combine two or more variables and model interdependences among the variables. In such models, the slope of one variable may depend on another variable. Partial effects are calculated as partial derivatives, similar to the polynomial equations we have already studied, considering the interaction term a a product of the variables involved.

* Real-world relationships are often not simply additive and can depend on multiple factors simultaneously.
* For example, the effect of exercise on health might depend on one's diet, illustrating that both factors together influence the outcome more intricately than separately.
* Potential pitfalls of ignoring interaction terms are biased estimates or overlooked relationships.

The data file *pizza4* includes 40 observations on an individual’s age and his or her spending on pizza. It is natural to believe that there is some connection between a person’s age and her pizza purchases, but also that this effect may depend on the person’s income. Equation 5.14 combines *age* and *income* in an interaction term, while Equation 5.15 shows the marginal effect of *age* on the expected pizza purchases.

*pizza* = *β*1 + *β*2*age* + *β*3*income* + *β*4(*age × income*) + *e* (5.14)

A math equation with black text

Description automatically generated with medium confidence(5.15)

Let us calculate the marginal effect of *age* on *pizza* for two levels of income, $25000 and $90000. There are two ways to introduce interaction terms in *R*; first, with a : (colon) symbol, when only the interaction term is created; second, with the \* (star) symbol when all the terms involving the variables in the interaction term should be

present in the regression. For instance, the code *x ∗ z* creates three terms: *x*, *z*, and *x* : *z* (this last one is *x* ‘interacted’ with *z*, which is a peculiarity of the *R* system).

**data**("pizza4",package="PoEdata")

mod3 <- **lm**(pizza~age\*income, data=pizza4)

**summary**(mod3)

##

## Call:

## lm(formula = pizza ~ age \* income, data = pizza4)

##

## Residuals:

## Min 1Q Median 3Q Max

## -200.9 -83.8 20.7 85.0 254.2

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|) ## (Intercept) 161.4654 120.6634 1.34 0.189

## age -2.9774 3.3521 -0.89 0.380

## income 6.9799 2.8228 2.47 0.018 \* ## age:income -0.1232 0.0667 -1.85 0.073 .

## ---

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

##

## Residual standard error: 127 on 36 degrees of freedom

## Multiple R-squared: 0.387, Adjusted R-squared: 0.336 ## F-statistic: 7.59 on 3 and 36 DF, p-value: 0.000468

inc <- **c**(25, 90)

b2 <- **coef**(mod3)[[2]]

b4 <- **coef**(mod3)[[4]]

(DpDa <- b2+b4\*inc)

## [1] -6.05841 -14.06896

The marginal effect of *age* is a reduction in pizza expenditure by $6*.*058407 for an income of $25000 and by $14*.*068965 for an income of $90000.

Equation 5.16 is another example of a model involving an interaction term: the *wage* equation. The equation involves, besides the interaction term, logs of the response (dependent) variable is in logs and a quadratic term.

*log*(*wage*) = *β*1 + *β*2*educ* + *β*3*exper* + *β*4(*educ × exper*) + *β*5*exper*2 + *e* (5.16)

Table 5.7: Wage equation with interaction and quadratic terms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Estimate | Std..Error | t.value | Pr...t.. |
| (Intercept) | 0.529677 | 0.226741 | 2.33604 | 0.019687 |
| educ | 0.127195 | 0.014719 | 8.64167 | 0.000000 |
| exper | 0.062981 | 0.009536 | 6.60447 | 0.000000 |
| I(exper^2) | -0.000714 | 0.000088 | -8.10911 | 0.000000 |
| educ:exper | -0.001322 | 0.000495 | -2.67222 | 0.007658 |

The marginal effect of an extra year of experience is determined, again, by the partial derivative of *log*(*wage*) with respect to *exper*, as Equation 5.17 shows. Here, however, the marginal effect is easier expressed in percentage change, rather than in units of *wage*.

A black and white text

Description automatically generated(5.17)

Please note that the marginal effect of *exper* depends on both education and experience. Let us calculate this marginal effect for the average values of *educ* and *exper*.

|  |
| --- |
| **data**(cps4\_small)  meduc <- **mean**(cps4\_small$educ)  mexper <- **mean**(cps4\_small$exper)  mod4 <- **lm**(**log**(wage)~educ\*exper+**I**(exper^2), data=cps4\_small)  smod4 <- **data.frame**(**xtable**(**summary**(mod4)))  b3 <- **coef**(mod4)[[3]]  b4 <- **coef**(mod4)[[4]]  b5 <- **coef**(mod4)[[5]]  pDwDex <- 100\*(b3+b4\*meduc+2\*b5\*mexper) |

The result of this calculation is %∆*wage* = *−*1*.*698, which has, apparently, the wrong sign. What could be the cause? Let us look at a summary of the regression output and identify the terms (see Table 5.7).

**kable**(smod4, caption="Wage equation with interaction and quadratic terms")

The output table shows that the order of the terms in the regression equation is not the same as in Equation 5.16, which is the effect of using the compact term *educ∗exper* in the *R* code. This places the proper interaction term, *educ* : *exper* in the last position of all terms containing any of the variables involved in the combined term *educ∗exper*. There are a few ways to solve this issue. One is to write the equation code in full, without the shortcut *educ ∗ exper*; another is to use the names of the terms when retrieving the coefficients, such as b4 <- coef(mod1)[["I(experˆ2)"]]; another to change the position of the terms in Equation 5.16 according to Table 5.7 and re-calculate the derivative in Equation 5.17. I am going to use the names of the variables, as they appear in Table 5.7, but I also change the names of the parameters to avoid confusion.

Table 5.8: Anova table for the basic \*andy\* model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum.Sq | Mean.Sq | F.value | Pr..F. |
| price | 1 | 1219.091 | 1219.0910 | 51.06310 | 0.000000 |
| advert | 1 | 177.448 | 177.4479 | 7.43262 | 0.008038 |
| Residuals | 72 | 1718.943 | 23.8742 | NA | NA |

|  |
| --- |
| bexper <- **coef**(mod4)[["exper"]]  bint <- **coef**(mod4)[["educ:exper"]]  bsqr <- **coef**(mod4)[["I(exper^2)"]]  pDwDex <- 100\*(bexper+bint\*meduc+2\*bsqr\*mexper) |

The new result indicates that the expected wage increases by 0*.*68829 percent when experience increases by one year, which, at least, has the “right” sign.

## 5.7 Goodness-of-Fit in Multiple Regression

The coefficient of determination, *R*2, has the same formula (Equation 5.18)and interpretation as in the case of the simple regression model.

A black text on a white background

Description automatically generated(5.18)

As we have already seen, the *R* function anova() provides a decomposition of the total sum of squares, showing by how much each predictor contributes to the reduction in the residual sum of squares. Let us look at the basic *andy* model described in Equation 5.2.

|  |
| --- |
| mod1 <- **lm**(sales~price+advert, data=andy)  smod1 <- **summary**(mod1) Rsq <- smod1$r.squared  anov <- **anova**(mod1)  dfr <- **data.frame**(anov)  **kable**(dfr, caption="Anova table for the basic \*andy\* model") |

Table 5.8 shows that *price* has the largest contribution to the reduction of variablity in the residuals; since total sum of squares is *SST* = 3115*.*482 (equal to the sum of column Sum.Sq in Table 5.8), the portion explained by *price* is 1219*.*091, the one explained by *advert* is 177*.*448, and the portion still remaining in the residuals is *SSE* = 1718*.*943. The portion of variability explained by regression is equal to the sum of the first two items in the Sum.Sq column of the anova table, those corresponding to *price* and *advert*: *SSR* = 1396*.*539.

RStudio displays the values of all named variables in the Environment window (the NE panel of your screen). Please check, for instance, that *R*2 (named Rsq in the previous code sequence) is equal to 0*.*448.

**Video 2 -** Further Inference in Multiple Regression

|  |
| --- |
| **rm**(list=**ls**())  **library**(PoEdata) *#for PoE datasets*  **library**(knitr) *#for referenced tables with kable()*  **library**(xtable) *#makes data frame for kable*  **library**(printr) *#automatically prints output nicely*  **library**(effects)  **library**(car)  **library**(AER)  **library**(broom) *#for tidy lm output and function glance()*  **library**(stats) |

New package: broom (Robinson 2016).

## 6.1 Joint Hypotheses and the F-statistic

A joint hypothesis is a set of relationships among regression parameters, relationships that need to be simultaneously true according to the null hypothesis. Joint hypotheses can be tested using the *F*-statistic that we have already met. Its formula is given by Equation 6.1. The *F*-statistic has an *F* distribution with the degrees of freedom *J* and *N − K*.

A black text on a white background

Description automatically generated(6.1)

In Equation 6.1 the subscript *U* stands for “unrestricted,” that is, the initial regression equation; the “restricted” equation is a new equation, obtained from the initial one, with the relationships in the null hypothesis assumed to hold. For example, if the initial equation is Equation 6.2 and the null hypothesis is Equation 6.3, then the restricted equation is Equation 6.4.

|  |  |
| --- | --- |
| *y* = *β*1 + *β*2*x*2 + *β*3*x*3 + *β*4*x*4 + *e* | (6.2) |
| *H*0 : *β*2 = 0 *AND β*3 = 0; *HA* : *β*2 ≠0 *OR β*3 ≠0 | (6.3) |
| *y* = *β*1 + *β*4*x*4 + *e* | (6.4) |

The symbol *J* in the *F* formula (Equation 6.1) is the first (*numerator*) degrees of freedom of the *F* statistic and is equal to the number of simultaneous restrictions in the null hypothesis (Equation 6.3); the second (the *denominator*) degrees of freedom of the *F*-statistic is *N −K*, which is the usual degrees of freedom of the unrestricted regression model (Equation 6.2). The practical procedure to test a joint hypothesis like the one in Equation 6.3 is to estimate the two regressions (unrestricted and restricted) and to calculate the *F*-statistic.

## 6.2 Testing Simultaneous Hypotheses

Let’s look, again, at the quadratic form of the *andy* equation (Equation 6.5).

*sales* = *β*1 + *β*2*price* + *β*3*advert* + *β*4*advert*2 + *e* (6.5)

Equation 6.5 has two terms that involve the regressor *advert*, of which at least one needs to be significant for a relationship between *advert* and *sales* to be established. To test if such a relationship exists, we can formulate the following test:

*H*0 : *β*3 = 0 *AND β*4 = 0; *HA* : *β*3 ≠ 0 *OR β*4 ≠ 0 (6.6)

I have already mentioned that *R* can do an *F* test quite easily (remember the function linearHypothesis?), but for learning purposes let us calculate the *F*-statistic in steps. The next code sequence uses information in the anova-type object, which, remember, can be visualized simply by typing the name of the object in the RStudio’s Console window.

alpha <- 0.05

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

N <- **NROW**(andy) *#Number of observations in dataset*

K <- 4 *#Four Betas in the unrestricted model*

J <- 2 *#Because Ho has two restrictions*

fcr <- **qf**(1-alpha, J, N-K)

mod1 <- **lm**(sales~price+advert+**I**(advert^2), data=andy)

anov <- **anova**(mod1) anov *# prints 'anova' table for the unrestricted model*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| price | 1 | 1219.091 | 1219.0910 | 56.49523 | 0.000000 |
| advert | 1 | 177.448 | 177.4479 | 8.22331 | 0.005441 |
| I(advert^2) | 1 | 186.858 | 186.8585 | 8.65941 | 0.004393 |
| Residuals | 71 | 1532.084 | 21.5787 | NA | NA |

|  |
| --- |
| SSEu <- anov[4, 2]  mod2 <- **lm**(sales~price, data=andy) *# restricted*  anov <- **anova**(mod2)  anov *# prints the 'anova' table for the restrictred model* |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Df | Sum Sq | Mean Sq | F value | Pr(>F) |
| price | 1 | 1219.09 | 1219.091 | 46.9279 | 0 |
| Residuals | 73 | 1896.39 | 25.978 | NA | NA |

|  |
| --- |
| SSEr <- anov[2,2]  fval <- ((SSEr-SSEu)/J) / (SSEu/(N-K))  pval <- 1-**pf**(fval, J, N-K) |

The calculated *F*-statistic is *fval* = 8*.*441 and the critical value corresponding to a significance level *α* = 0*.*05 is 3*.*126, which rejects the null hypothesis that both *β*3 and *β*4 are zero. The *p*-value of the test is *p* = 0*.*0005.

Using the linearHypothesis() function should produce the same result:

Hnull <- **c**("advert=0", "I(advert^2)=0")

**linearHypothesis**(mod1,Hnull)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| 73 | 1896.39 | NA | NA | NA | NA |
| 71 | 1532.08 | 2 | 364.306 | 8.44136 | 0.000514 |

The table generated by the linearHypothesis() function shows the same values of the *F*-statistic and *p*-value that we have calculated before, as well as the residual sum of squares for the restricted and unrestricted models. Please note how I formulate the joint hypothesis as a vector of character values in which the names of the variables perfectly match those in the unrestricted model.

Testing the overall significance of a model amounts to testing the joint hypothesis that all the slope coefficients are zero. *R* does automatically this test and the resulting *F*-statistic and *p*-value are reported in the regression output.

**summary**(mod1)

##

## Call:

## lm(formula = sales ~ price + advert + I(advert^2), data = andy)

##

## Residuals:

## Min 1Q Median 3Q Max

## -12.255 -3.143 -0.012 2.851 11.805

##

## Coefficients:

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

## (Intercept) 109.719 6.799 16.14 < 2e-16 \*\*\* ## price -7.640 1.046 -7.30 3.2e-10 \*\*\*

## advert 12.151 3.556 3.42 0.0011 \*\* ## I(advert^2) -2.768 0.941 -2.94 0.0044 \*\*

## ---

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

##

## Residual standard error: 4.65 on 71 degrees of freedom

## Multiple R-squared: 0.508, Adjusted R-squared: 0.487

## F-statistic: 24.5 on 3 and 71 DF, p-value: 5.6e-11

The *F*-statistic can be retrieved from summary(mod1) or by using the function glance(modelname) in package broom, as shown in the following code lines. The function tidy, also from package broom organizes regression output (mainly the coefficients and their statistics) in a neat table. Both glance and tidy create output in the form of data.frame, which makes it suitable for use by other functions such as kable and ggplot2. Please also note that tidy(mod1) and tidy(summary(mod1)) produce the same result, as shown in Tables 6.1 and 6.2. As always, we can use the function names to obtain a list of the quantities available in the output of the glance function.

smod1 <- **summary**(mod1)

**kable**(**tidy**(smod1), caption="Tidy 'summary(mod1)' output")

fval <- smod1$fstatistic

Table 6.1: Tidy ’summary(mod1)’ output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 109.71904 | 6.799045 | 16.13742 | 0.000000 |
| Price | -7.64000 | 1.045939 | -7.30444 | 0.000000 |
| Advert | 12.15124 | 3.556164 | 3.41695 | 0.001052 |
| I(advert^2) | -2.76796 | 0.940624 | -2.94269 | 0.004393 |

Table 6.2: ’Tidy(mod1)’ output

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 109.71904 | 6.799045 | 16.13742 | 0.000000 |
| Price | -7.64000 | 1.045939 | -7.30444 | 0.000000 |
| Advert | 12.15124 | 3.556164 | 3.41695 | 0.001052 |
| I(advert^2) | -2.76796 | 0.940624 | -2.94269 | 0.004393 |

**library**(broom)

**kable**(**tidy**(mod1), caption="'Tidy(mod1)' output")

**glance**(mod1)$statistic *#Retrieves the F-statistic*

## [1] 24.4593

**names**(**glance**(mod1)) *#Shows what is available in 'glance'*

## [1] "r.squared" "adj.r.squared" "sigma" "statistic"

## [5] "p.value" "df" "logLik" "AIC"

## [9] "BIC" "deviance" "df.residual"

**kable**(**glance**(mod1),

caption="Function 'glance(mod1)' output", digits=2,

col.names=(**c**("Rsq","AdjRsq","sig","F", pF","K","logL","AIC","BIC","dev","df.res")))

Table 6.3 shows a summary of the quadratic *andy* model (mod1), where I have changed the names of various items so that the table fits the width of the page.

Table 6.3: Function ’glance(mod1)’ output

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Rsq | AdjRsq | sig | F | pF | K | logL | AIC | BIC | dev | df.res |
| 0.51 | 0.49 | 4.65 | 24.46 | 0 | 4 | -219.55 | 449.11 | 460.7 | 1532.08 | 71 |

Table 6.4: Joint hypotheses with the ’linearHypothesis’ function

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| res.df | rss | df | sumsq | statistic | p.value |
| 73 | 1779.86 | NA | NA | NA | NA |
| 71 | 1532.08 | 2 | 247.776 | 5.74123 | 0.004885 |

When retrieving these variables, make sure you use the original names as indicated by the names(glance(mod1)) command.

When testing a two-tail single (not joint) null hypothesis, the *t* and *F* tests are equivalent. However, one-tail tests, single or joint cannot be easily performed by an *F* test.

Let us solve one more exercise involving a joint hypothesis with linear combinations of regression coefficients. Suppose we want to test the simultaneous hypotheses that the monthly advertising expenditure *advert*0 = $1900 in the quadratic *andy* model (Equation 6.5) satisfies the profit-maximizing condition *β*3 + 2*β*4*advert*0 = 1 , and that, when *price* = $6 and *advert* = $1900 sales revenue is $80000.

|  |
| --- |
| hyp <- **c**("advert+3.8\*I(advert^2)=1",  "(Intercept)+6\*price+1.9\*advert+3.61\*I(advert^2)=80")  lhout <- **tidy**(**linearHypothesis**(mod1,hyp))  **kable**(lhout, caption="Joint hypotheses with the 'linearHypothesis' function") |

Table 6.4 includes the *F*-statistic of the test, *F* = 5*.*741229 and its *p*-value *p* = 0*.*004885. Please be aware that the function tidy changes the names in the output of linearHypothesis. A useful exercise is to compare the raw output of linearHypothesis with the output generated by tidy(linearHypothesis(mod1)). There are several other possibilities to compare two regression models, such as a restricted and unrestricted ones in *R*, such as the anova() function or Wald tests.

These are going to be mentioned in later chapters.

Lab 1 - Omitted Variable Bias

Consider the general model with two regressors in Equation 6.7, a model that I will call the *true* model.

*y* = *β*1 + *β*2*x*2 + *β*3*x*3 + *e* (6.7)

Suppose we are only interested in estimating *β*2, but there is no data available for *x*3, or for other reasons *x*3 is omitted from the model in Equation 6.7. What is the error in the estimate of *β*2 introduced by omitting *x*3? Equation 6.8 shows what is left of the *true* model after omitting *x*3.

Table 6.5: The correct model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -5533.63 | 11229.533 | -0.492775 | 0.622426 |
| he | 3131.51 | 802.908 | 3.900209 | 0.000112 |
| we | 4522.64 | 1066.327 | 4.241328 | 0.000027 |

*y* = *β*1 + *β*2*x*2 + *u* (6.8)

Let *b∗*2 be the estimate of *β*2 when *x*3 is omitted. Equation 6.9 gives the bias in this simple, two-regressor case. The formula shows that bias depends on the direct relationship between the omitted regressor and response through *β*3, as well as the correlation between the omitted and the included regressors. When *β*3 and *cov*(*x*2*, x*3) are both positive or both negative the bias is positive (the incorrect model overestimates the true *β*2), and when they are of opposite signs the bias is negative (*β*2 is underestimated)

A math equation with numbers and symbols

Description automatically generated with medium confidence(6.9)

The example in this section uses the dataset *edu*\_*inc*, and two models: one where the response variable *family income* (*faminc*) is explained by the regressors *husband’s education* (*he*) and *wife’s education* (*we*), and another model, where the *we* regressor is omitted. The purpose is to compare the estimates coming from the two models and see if there is a significant difference between them.

**data**("edu\_inc", package="PoEdata")

mod1 <- **lm**(faminc~he+we, data=edu\_inc)

mod2 <- **lm**(faminc~he, data=edu\_inc)

**kable**(**tidy**(mod1), caption="The correct model")

**kable**(**tidy**(mod2), caption="The incorrect model ('we' omitted)")

The marginal effect of husband’s education is much lower in the incorrect model. Let us apply the logic of Equation 6.9 to the *edu*\_*inc* model. The direct effect of the omitted regressor (*we*) on response (*faminc*) is likely to be positive in theory (higher education generates higher income); the correlation between husband’s and wife’s education is also likely to be positive if we believe that people generally marry persons within their entourage. Thus, we should expect that omitting the regressor *we* should produce an overestimated marginal effect of *he*. Our data happen to confirm this supposition, though there is some chance that they might not.

Table 6.6: The incorrect model (’we’ omitted)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 26191.27 | 8541.108 | 3.06650 | 0.002304 |
| he | 5155.48 | 658.457 | 7.82964 | 0.000000 |

Table 6.7: Correct ’faminc’ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | -7755.33 | 11162.935 | -0.694739 | 0.487599 |
| He | 3211.53 | 796.703 | 4.031022 | 0.000066 |
| We | 4776.91 | 1061.164 | 4.501574 | 0.000009 |
| kl6 | -14310.92 | 5003.928 | -2.859937 | 0.004447 |

Understanding the problem of omitted variable is very important because it can justify the choice of a particular model. If one is not interested in the effect of variable *x*3 on *y* and can convince that *x*3 is uncorrelated with *x*2, one can argue with criticism about omitting the important regressor *x*3.

## 6.4 Irrelevant Variables

We have seen the effect of omitting a relevant regressor (the effect is biased estimates and lower variances of the included regressors). But what happens if irrelevant variables are incorrectly included? Not surprisingly, this increases the variances (lowers the precision) of the other variables in the model. The next example uses the same (*edu*\_*inc*) dataset as above, but includes two artificially generated variables, *xtra*\_*x*5 and *xtra*\_*x*6 that are correlated with *he* and *we* but, obviously, have no role in determining *y*. Let us compare two models, of which one includes these irrelevant variables.

mod3 <- **lm**(faminc~he+we+kl6, data=edu\_inc) mod4 <- **lm**(faminc~he+we+kl6+xtra\_x5+xtra\_x6, data=edu\_inc) **kable**(**tidy**(mod3), caption="Correct 'faminc' model")

**kable**(**tidy**(mod4),

caption="Incorrect 'faminc' with irrelevant variables")

Table 6.8: Incorrect ’faminc’ with irrelevant variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | -7558.613 | 11195.41 | -0.675153 | 0.499948 |
| He | 3339.792 | 1250.04 | 2.671750 | 0.007838 |
| We | 5868.677 | 2278.07 | 2.576165 | 0.010329 |
| kl6 | -14200.184 | 5043.72 | -2.815419 | 0.005100 |
| xtra\_x5 | 888.843 | 2242.49 | 0.396364 | 0.692037 |
| xtra\_x6 | -1067.186 | 1981.69 | -0.538524 | 0.590499 |

A comparison of the two models shown in Tables 6.7 and 6.8 indicates that the inclusion of the two irrelevant variables has increased the marginal effects, standard errors, and the *p*-values of *he* and *we*. Thus, including irrelevant variables may incorrectly diminish the significance of the “true” regressors.

## 6.5 Model Selection Criteria

The main tools of building a model should be economic theory, sound reasoning based on economic principles, and making sure that the model satisfies the Gauss-Markov assumptions. One should also consider the possibility of omitted variable bias and the exclusion of irrelevant variables that may increase variablility in the estimates. After all these aspects have been considered and a model established, there are a few quantities that help comparing different models. These are *R*2, adjusted *R*2 (*R*¯2), the Akaike information criterion (*AIC*), and the Schwarz (or Bayesian information) criterion (*SC* or *BIC*).

We have already seen how to calculate the coefficient of determination, *R*2 and how it measures the distance between the regression line and the observation points. A major disadvantage of *R*2 is that it increases every time a new regressor is included in the model, whether the regressor is relevant or not. The idea of counterbalancing this property of *R*2 has lead to a new measure, adjusted *R*2, denoted by *R*¯2, given by Equation 6.10.

A close-up of a line

Description automatically generated (6.10)

Adjusted *R*2, while addressing the problem with *R*2, introduces other problems. In general, no single goodness of fit measure is perfect. The Akaike information criterion (*AIC*) and the Schwarz criterion use the same idea of penalizing the introduction of extra regressors. Their formulas are given in Equations 6.11 and 6.12.

A black and white math symbol

Description automatically generated (6.11)

A math equation with black text

Description automatically generated with medium confidence (6.12)

Among several models, the best fit is the one that maximizes *R*2 or *R*¯2. On the contrary, the best model must **minimize** *AIC* or *BIC*. Some computer packages, *R* included, calculate *AIC* and *BIC* differently than Equations 6.11 and 6.12 indicate. However, the ranking of the various models is the same.

The following code sequence needs some explanation. Function as.numeric extracts only the numbers from an object such as glance(mod1), which also contains row and column names. The purpose is to put together a table with information comming from several models. Function rbind gathers several rows in a matrix, which then is made into a data.frame and given the name tab. The part of code [,c(1,2,8,9)] at the end of rbind instructs *R* to pick all rows, but only columns 1, 2, 8, and 9 from the glance table. Function row.names assigns or changes the row names in a data frame; finally, kable, which we have already encountered several times, prints the table, assigns column names, and gives a caption to the table. While there are many ways to create a table in *R*, I use kable from package knitr because it allows me to cross-reference tables within this book. kable only works with data frames.

|  |
| --- |
| mod1 <- **lm**(faminc~he, data=edu\_inc)  mod2 <- **lm**(faminc~he+we, data=edu\_inc)  mod3 <- **lm**(faminc~he+we+kl6, data=edu\_inc)  mod4 <- **lm**(faminc~he+we+kl6+xtra\_x5+xtra\_x6, data=edu\_inc)  r1 <- **as.numeric**(**glance**(mod1))  r2 <- **as.numeric**(**glance**(mod2))  r3 <- **as.numeric**(**glance**(mod3))  r4 <- **as.numeric**(**glance**(mod4))  tab <- **data.frame**(**rbind**(r1, r2, r3, r4))[,**c**(1,2,8,9)]  **row.names**(tab) <- **c**("he","he, we","he, we, kl6","he, we, kl6, xtra\_x5, xtra\_x6")  **kable**(tab, caption="Model comparison, 'faminc' ", digits=4, col.names=**c**("Rsq","AdjRsq","AIC","BIC")) |

Table 6.9 shows the four model selection criteria for four different models based on the *edu*\_*inc* dataset, with the first column showing the variables included in each model. It is noticeable that three of the criteria indicate the third model as the best fit, while one, namely *R*2 prefers the model that includes the irrelevant variables *xtra*\_*x*5 and *xtra*\_*x*6.

Table 6.9: Model comparison, ’faminc’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Rsq | AdjRsq | AIC | BIC |
| He | 0.1258 | 0.1237 | 10316.7 | 10328.8 |
| he, we | 0.1613 | 0.1574 | 10300.9 | 10317.1 |
| he, we, kl6 | 0.1772 | 0.1714 | 10294.7 | 10315.0 |
| he, we, kl6, xtra\_x5, xtra\_x6 | 0.1778 | 0.1681 | 10298.4 | 10326.8 |

As a side note, a quick way of extracting the information criteria from an lm() object is illustrated in the following code fragment.

**library**(stats)

smod1 <- **summary**(mod1)

Rsq <- smod1$r.squared

AdjRsq <- smod1$adj.r.squared

aic <- **AIC**(mod1)

bic <- **BIC**(mod1)

**c**(Rsq, AdjRsq, aic, bic)

## [1] 0.125801 0.123749 10316.651535 10328.828905

Another potentially useful tool for building an appropriate model is the Ramsey specification test, RESET. This method automatically adds higher-order polynomial terms to your model and tests the joint hypothesis that their coefficients are all zero. Thus, the null hypothesis of the test is *H*0: “No higher-order polynomial terms are necessary”; if we reject the null hypothesis we need to consider including such terms.

The *R* function that performs a RESET test is resettest, which requires the following argumets: formula, the formula of the model to be tested or the name of an already calculated lm object; power, a set of integers indicating the powers of the polynomial terms to be included; type, which could be one of “fitted”, “regressor”, or “princomp”, indicating whether the aditional terms should be powers of the regressors, fitted values, or the first principal component of the regressor matrix; and, finally, data, which specifies the dataset to be used if a formula has been provided and not a model object. The following code applies the test to the complete *faminc* model, first using only quadratic terms of the fitted values, then using both quadratic and cubic terms.

mod3 <- **lm**(faminc~he+we+kl6, data=edu\_inc)

**resettest**(mod3, power=2, type="fitted")

##

## RESET test

##

Table 6.10: A simple linear ’mpg’ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 42.91551 | 0.834867 | 51.4040 | 0 |
| cyl | -3.55808 | 0.145676 | -24.4247 | 0 |

## data: mod3

## RESET = 5.984, df1 = 1, df2 = 423, p-value = 0.0148

**resettest**(mod3, power=2:3, type="fitted")

##

## RESET test

##

## data: mod3

## RESET = 3.123, df1 = 2, df2 = 422, p-value = 0.0451

The number labeled as RESET in the output is the *F*-statistic of the test under the null hypothesis followed by the two types of degrees of freedom of the *F* distribution and the *p*-value. In our case both *p*-values are slightly lower than 0*.*05, indicating that the model marginally fails the specification test and some higher order terms may be necessary.

Lab 2 - Collinearity

There is collinearity among regressors when two or more regressors move closely with each other or display little variability. A consequence of collinearity is large variance in the estimated parameters, which increases the chances of not finding them significantly different from zero. The estimates are, however, unbiased since

(imperfect) collinearity does not technically violate the Gauss-Markov assumptions. Collinearity tends to show insignificant coefficients even when measures of goodnessof-fit such as *R*2 or overall significance (the *F*-statistic) may be quite large.

Let us consider the example of the dataset *cars*, where *mpg* is miles per gallon, *cyl* is number of cylinders, *eng* is engine displacement in cubic inches, and *wgt* is the weight of the vehicle in pounds.

|  |
| --- |
| **data**("cars", package="PoEdata") mod1 <- **lm**(mpg~cyl, data=cars)  **kable**(**tidy**(mod1), caption="A simple linear 'mpg' model") |

This naive model suggests a strong effect of the number of cylinders on fuel economy, but if we introduce more terms in the equation this result changes substantially.

Table 6.11: Multivariate ’mpg’ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 44.370962 | 1.480685 | 29.966509 | 0.000000 |
| cyl | -0.267797 | 0.413067 | -0.648313 | 0.517166 |
| eng | -0.012674 | 0.008250 | -1.536225 | 0.125298 |
| wgt | -0.005708 | 0.000714 | -7.995143 | 0.000000 |

Table 6.12: Variance inflation factors for the ’mpg’ regression model

|  |  |
| --- | --- |
| regressor | VIF |
| Cyl | 10.51551 |
| Eng | 15.78646 |
| Wgt | 7.78872 |

mod2 <- **lm**(mpg~cyl+eng+wgt, data=cars)

**kable**(**tidy**(mod2), caption="Multivariate 'mpg' model")

In the model summarized in Table 6.11 the number of cylinders becomes insignificant alltogether, a sharp change with respect to the previous specification. This high sensitivity of the estimates when other variables are introduced is also a sign of collinearity. Indeed, it is reasonable to believe that the characteristics of the vehicles vary together: heavier vehicles have more cylinders and bigger engines.

A test that may be useful in detecting collinearity is to calculate the *variance inflation factor*, *V IF*, for each regressor. The rule of thumb is that a regressor produces collinearity if its *V IF* is greater than 10. Equation 6.13 shows the formula for the variance inflation factor, where *Rk*2 is the *R*2 from regressing the variable *xk* on all the remaining regressors.

1

*V IFk* = 1 *− Rk*2 (6.13)

|  |
| --- |
| mod2 <- **lm**(mpg~cyl+eng+wgt, data=cars) tab <- **tidy**(**vif**(mod2)) **kable**(tab,  caption="Variance inflation factors for the 'mpg' regression model", col.names=**c**("regressor","VIF")) |

The results in Table 6.12 show that the regressors cyl and eng fail the collinearity test, having *V IF*s greater than 10.

Table 6.13: Forecasting in the quadratic ’andy’ model

|  |  |  |
| --- | --- | --- |
| fit | lwr | upr |
| 76.974 | 67.5326 | 86.4155 |

## 6.7 Prediction and Forecasting

We have previously discussed the semantic difference between prediction and forecasting, with prediction meaning the estimation of an expected value of the response and forecasting meaning an estimate of a particular value of the response. We mentioned that, for the same vector of regressors, prediction has a narrower confidence interval than forecasting because forecasting includes, besides the uncertainty of the expected value of the response, the variablility of a particular observation about its mean. I essentially repeat the same procedure here for the quadratic *andy* model, which regresses *sales* on *price*, *advert*, and *advert*2. The key *R* function to calculate both predictions and forecasts is the function predict, with the following arguments: model, which is the name of a model object; newdata, which contains the new data points where prediction is desired; if newdata is missing, predictions are calculated for all observations in the dataset; interval, which can be “none”, “confidence”, or

“prediction”, and tells *R* whether we want only a point estimate of the response, a prediction with its confidence interval, or a forecast with its confidence interval; level, which is the confidence level we want; if missing, level is 95%; other arguments (see ?predict() for more details).

|  |
| --- |
| predpoint <- **data.frame**(price=6, advert=1.9) mod3 <- **lm**(sales~price+advert+**I**(advert^2), data=andy) **kable**(**tidy**(**predict**(mod3, newdata=predpoint, interval="prediction")),  caption="Forecasting in the quadratic 'andy' model") |

Table 6.13 displays the point estimate and forecast interval estimate for the data point *price* = $6 and *advert* = 1*.*9, which, remember, stands for advertising expenditure of $1900.

## Using Indicator Variables

|  |
| --- |
| **rm**(list=**ls**())  **library**(bookdown)  **library**(PoEdata)*#for PoE datasets*  **library**(knitr) *#for referenced tables with kable()*  **library**(xtable) *#makes data frame for kable*  **library**(printr) *#automatically prints output nicely*  **library**(effects)  **library**(car)  **library**(AER)  **library**(broom) *#for tidy lm output and function glance()*  **library**(stats)  **library**(lmtest)*#for coeftest() and other test functions*  **library**(stargazer) *#nice and informative tables* |

This chapter introduces the package lmtest (Hothorn et al. 2015).

## 7.1 Factor Variables

Indicator variables show the presence of a certain non-quantifiable attribute, such as whether an individual is male or female, or whether a house has a swimming pool. In *R*, an indicator variable is called *factor*, *category*, or *ennumerated type* and there is no distinction between binary (such as yes-no, or male-female) or multiple-category factors.

Factors can be either numerical or character variables, ordered or not. *R* automatically creates dummy variables for each category within a factor variable and excludes a

99

Table 7.1: Summary for ’utown’ dataset

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | price | sqft | age | utown | pool | fplace |
|  | Min. :134 | Min. :20.0 | Min. : 0.00 | 0:481 | 0:796 | 0:482 |
|  | 1st Qu.:216 | 1st Qu.:22.8 | 1st Qu.: 3.00 | 1:519 | 1:204 | 1:518 |
|  | Median :246 | Median :25.4 | Median : 6.00 | NA | NA | NA |
|  | Mean :248 | Mean :25.2 | Mean : 9.39 | NA | NA | NA |
|  | 3rd Qu.:278 | 3rd Qu.:27.8 | 3rd Qu.:13.00 | NA | NA | NA |
|  | Max. :345 | Max. :30.0 | Max. :60.00 | NA | NA | NA |

(baseline) category from a model. The choice of the baseline category, as well as which categories should be included can be changed using the function contrasts().

The following code fragment loads the utown data, declares the indicator variables utown, pool, and fplace as factors, and displays a summary statistics table (Table

7.1). Please notice that the factor variables are represented in the summary table as count data, i.e., how many observations are in each category.

**library**(stargazer);

**library**(ggplot2)

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

utown$utown <- **as.factor**(utown$utown)

utown$pool <- **as.factor**(utown$pool)

utown$fplace <- **as.factor**(utown$fplace)

**kable**(**summary.data.frame**(utown), caption="Summary for 'utown' dataset")

This example uses the utown dataset, where prices are in thousands, sqft is the living area is hundreds of square feet, and utown marks houses located near university.

## 7.2 Examples

mod4 <- **lm**(price~utown\*sqft+age+pool+fplace, data=utown)

**kable**(**tidy**(mod4), caption="The 'house prices' model")

bsqft <- 1000\***coef**(mod4)[["sqft"]]

bsqft1 <- 1000\*(**coef**(mod4)[["sqft"]]+**coef**(mod4)[["utown1:sqft"]])

Notice how the model in the above code sequence has been specified: the term utown\*sqft created three terms: utown, sqft, and the interaction term utown:sqft.

Table 7.2 shows the estimated coefficients and their statistics; please notice how the factor variables have been market at the end of their names with 1 to show which

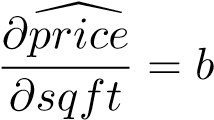
Table 7.2: The ’house prices’ model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 24.499985 | 6.191721 | 3.95689 | 0.000081 |
| utown1 | 27.452952 | 8.422582 | 3.25945 | 0.001154 |
| Sqft | 7.612177 | 0.245176 | 31.04775 | 0.000000 |
| Age | -0.190086 | 0.051205 | -3.71229 | 0.000217 |
| pool1 | 4.377163 | 1.196692 | 3.65772 | 0.000268 |
| fplace1 | 1.649176 | 0.971957 | 1.69676 | 0.090056 |
| utown1:sqft | 1.299405 | 0.332048 | 3.91331 | 0.000097 |

category they represent. For instance, utown1 is the equivalent of a dummy variable equal to 1 when utown is equal to 1. When a factor has *n* categories the regression output will show *n −* 1 dummies marked to identify each category.

According to the results in Table 7.2, an extra hundred square feet increases the price by bsqft=$7612*.*18 if the house is not near university and by bsqft1 = $8911*.*58 if the house is near university. This example shows how interaction terms allow distinguishing the marginal effect of a continuous variable (sqft) for one category (utown=1) from the marginal effect of the same continuous variable within another category (utown=0).

In general, the marginal effect of a regressor on the response is, as we have seen before, the partial derivative of the response with respect to that regressor. For example, the marginal effect of sqft in the *house prices* model is as shown in Equation 7.1.

[*sqft*] + *utown × b*[*utown* : *sqft*] (7.1)

Let us look at another example, the *wage* equation using the dataset *cps*4\_*small*.

**data**("cps4\_small", package="PoEdata")

**names**(cps4\_small)

## [1] "wage" "educ" "exper" "hrswk" "married" "female" "metro"

## [8] "midwest" "south" "west" "black" "asian"

This dataset already includes dummy variables for gender, race, and region, so that we do not need *R*’s capability of building these dummies for us. Although usually it is useful to declare the categorical variables as factors, I will not do so this time. Let us consider the model in Equation 7.2.

*wage* = *β*1 + *β*2*educ* + *δ*1*black* + *δ*2*female* + *γ*(*black × female*) (7.2)

Table 7.3: A wage-discrimination model

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | -5.28116 | 1.900468 | -2.77887 | 0.005557 |
| Educ | 2.07039 | 0.134878 | 15.35009 | 0.000000 |
| Black | -4.16908 | 1.774714 | -2.34916 | 0.019011 |
| Female | -4.78461 | 0.773414 | -6.18635 | 0.000000 |
| black:female | 3.84429 | 2.327653 | 1.65158 | 0.098937 |
| mod5 <- **lm**(wage~educ+black\*female, data=cps4\_small)  delta1 <- **coef**(mod5)[["black"]]  delta2 <- **coef**(mod5)[["female"]]  gamma <- **coef**(mod5)[["black:female"]]  blfm <- delta1+delta2+gamma  **kable**(**tidy**(mod5), caption="A wage-discrimination model") | | | | |

What are the expected wages for different categories, according to Equation 7.2 and Table 7.3?

* white male: *β*1 + *β*2*educ* (the baseline category)
* black male: *β*1 + *β*2*educ* + *δ*1
* white female: *β*1 + *β*2*educ* + *δ*2
* black female: *β*1 + *β*2*educ* + *δ*1 + *δ*2 + *γ*

These formulas help evaluating the differences in wage expectations between different categories. For instance, given the same education, the difference between black female and white male is *δ*1 + *δ*2 + *γ* = *−*5*.*11, which means that the average black female is paid less by $5*.*11 than the average white man.

To test the hypothesis that neither race nor gender affects wage is to test the joint hypothesis *H*0 : *δ*1 = 0*, δ*2 = 0*, γ* = 0 against the alterntive that at least one of these coefficients is different from zero. We have learned already how to perform an *F*-test the hard way and how to retrieve the quantities needed for the *F*-statistic (*SSER, SSEU, J*, and *N − K*). Let us this time use *R*’s linearHypothesis function.

|  |
| --- |
| hyp <- **c**("black=0", "female=0", "black:female=0")  tab <- **tidy**(**linearHypothesis**(mod5, hyp))  **kable**(tab, caption="Testing a joint hypothesis for the 'wage' equation") |

The results in Table 7.4 indicates that the null hypothesis of no discrimination can be rejected.

Table 7.4: Testing a joint hypothesis for the ’wage’ equation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| res.df | rss | df | sumsq | statistic | p.value |
| 998 | 135771 | NA | NA | NA | NA |
| 995 | 130195 | 3 | 5576.47 | 14.2059 | 0 |

Lab 3 - Comparing Two Regressions: the Chow Test

By interacting a binary indicator variable with all the terms in a regression and testing that the new terms are insignificant we can determine if a certain category is significantly different than the other categories. Starting with the *wage* regression in Equation 7.2, let us include the indicator variable “south”.

|  |
| --- |
| dnosouth <- cps4\_small[**which**(cps4\_small$south==0),]*#no south*  dsouth <- cps4\_small[**which**(cps4\_small$south==1),] *#south*  mod5ns <- **lm**(wage~educ+black\*female, data=dnosouth)  mod5s <- **lm**(wage~educ+black\*female, data=dsouth)  mod6 <- **lm**(wage~educ+black\*female+south/(educ+black\*female), data=cps4\_small)  **stargazer**(mod6, mod5ns, mod5s, header=FALSE, type=.stargazertype,  title="Model comparison, 'wage' equation", keep.stat="n",digits=2, single.row=TRUE, intercept.bottom=FALSE) |

The table titled “Model comparison, ‘wage’ equation” presents the results of three equations: equation (1) is the full *wage* model with all terms interacted with variable *south*; equation (2) is the basic ‘wage’ model shown in Equation 7.2 with the sample restricted to non-south regions, and, finally, equation (3) is the same as (2) but with the sample restricted only to the south region. (This table is constructed with the function stargazer from the package by the same name (Hlavac 2015).)

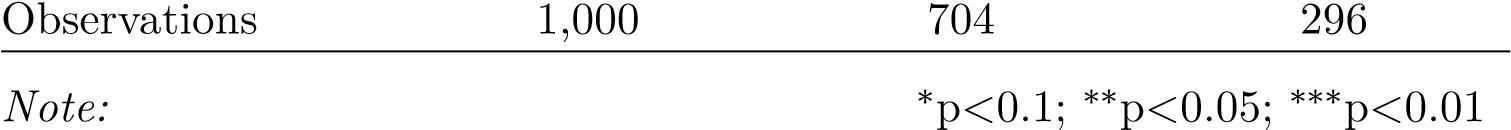
However, in *R* it is not necessary to split the data manually as I did in the above code sequence; instead, we just write two equations like mod5 and mod6 and ask *R* to do a Chow test to see if the two equations are statistically identical. The Chow test is performed in *R* by the function anova, with the results presented in Table 7.6, where the *F* statistic is equal to 0*.*320278 and the corresponding *p*-value of 0*.*900945.

**kable**(**anova**(mod5, mod6), caption="Chow test for the 'wage' equation")

Table 7.6 indicates that the null hypothesis that the equations are equivalent cannot be rejected. In other words, our test provides no evidence that wages in the south Table 7.5: region are statistically different from the rest of the regions.

Model comparison, ’wage’ equation

*Dependent variable:*

wage

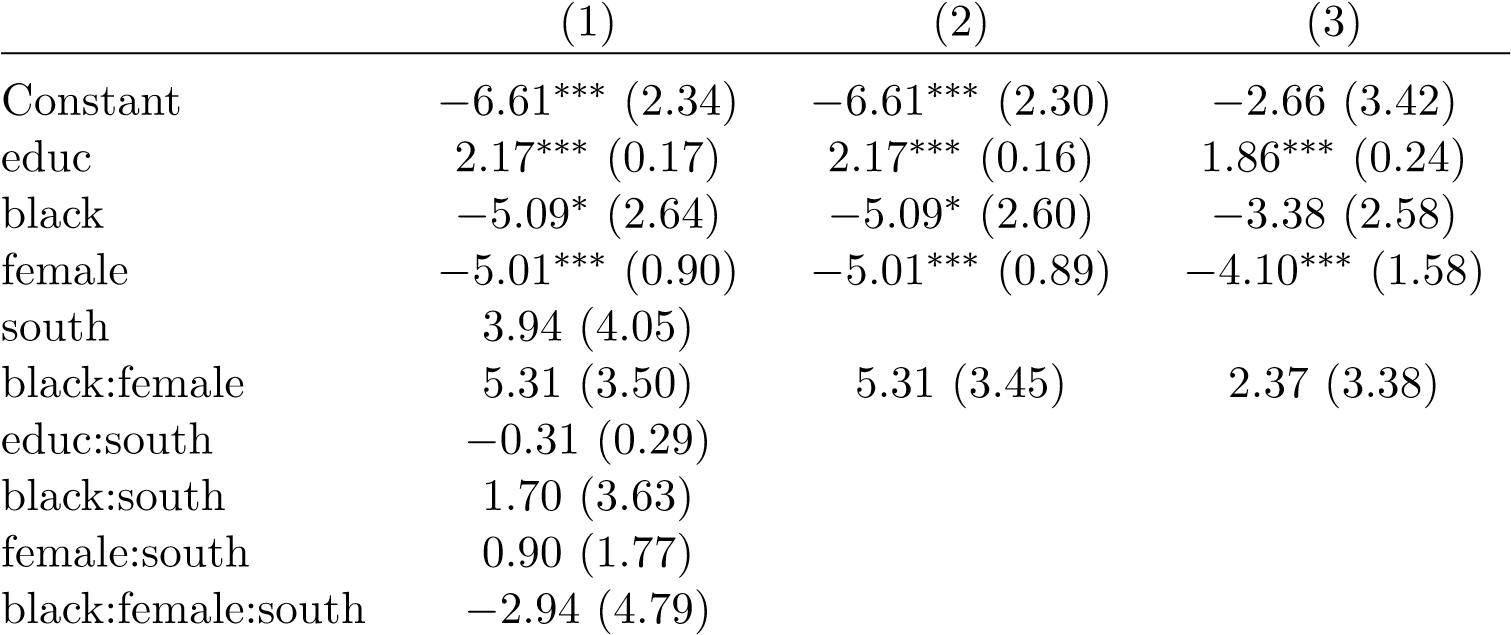


Table 7.6: Chow test for the ’wage’ equation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| 995 | 130195 | NA | NA | NA | NA |
| 990 | 129984 | 5 | 210.258 | 0.320278 | 0.900945 |

## 

## Indicator Variables in Log-Linear Models

An indicator regressor is essentially no different from a continuous one and its interpretation is very similar to the one we have studied in the context of log-linear models. In a model like *log*(*y*) = *β*1+*β*2*x*+*δD*, where *D* is an indicator variable, the percentage difference between the two categories represented by *D* can be calculated in two ways, both using the coefficient *δ*:

* approximate: %∆*y ∼*= 100*δ*
* exact: %∆*y* = 100(*eδ −* 1)

Let us calculate these two effets in a log-linear wage equation based on the dataset *cps*4\_*small*.

**data**("cps4\_small", package="PoEdata") mod1 <- **lm**(**log**(wage)~educ+female, data=cps4\_small) approx <- 100\***coef**(mod1)[["female"]] exact <- 100\*(**exp**(**coef**(mod1)[["female"]])-1)

The results indicate a percentage difference in expected wage between females and males as follows: %∆*wage* = *−*24*.*32% (approximate), or %∆*wage* = *−*21*.*59%

(exact).

## 7.5 The Linear Probability Model

Linear probability models are regression models in which the response, rather than a regressor is a binary indicator variable. However, since the regressors can be either continuous or factor variables, the fitted values will be continuous. Equation 7.3, where *y ∈ {*0*,*1*}* shows a general linear probability model.

*y* = *β*1 + *β*2*x*2 + *...* + *βkxk* + *e* (7.3)

How can a continous fitted variable be interpreted when the actual response is binary? As it turns out, the fitted value represents the *probability* that the response takes the value 1, *y*ˆ = *Pr*(*y* = 1). There are two major problems with this model. First, the model is heteroskedastic, with the variance being larger in the middle range of the fitted values; second, the fitted variable being continuous, it can take values, unlike a probability function, outside the interval [0*,*1].

Table 7.7: Linear probability model, the ’coke’ example

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.890215 | 0.065485 | 13.59421 | 0.000000 |
| pratio | -0.400861 | 0.061349 | -6.53407 | 0.000000 |
| disp\_coke | 0.077174 | 0.034392 | 2.24397 | 0.025026 |
| disp\_pepsi | -0.165664 | 0.035600 | -4.65352 | 0.000004 |

The next example, based on the dataset *coke* estimates the probability that a customer chooses *coke* when *coke* or *pepsi* are displayed. Besides the two indicator variables *disp*\_*coke* and *disp*\_*pepsi*, the model includes the price ratio between *coke* and *pepsi*.

|  |
| --- |
| *# Linear probability example* **data**("coke", package="PoEdata")  mod2 <- **lm**(coke~pratio+disp\_coke+disp\_pepsi, data=coke) **kable**(**tidy**(mod2),  caption="Linear probability model, the 'coke' example") |
| *# Graph for the linear probability model* b00 <- **coef**(mod2)[[1]]  b10 <- b00+**coef**(mod2)[["disp\_coke"]] b11 <- b10+**coef**(mod2)[["disp\_pepsi"]] b01 <-b11-**coef**(mod2)[["disp\_coke"]] b2 <- **coef**(mod2)[["pratio"]] **plot**(coke$pratio, coke$coke, ylab="Pr[coke]", xlab="price ratio")  **abline**(b00, b2, lty=2, col=2) **abline**(b10,b2, lty=3, col=3) **abline**(b11,b2, lty=4, col=4) **abline**(b01,b2, lty=5, col=5)  **legend**("topright", **c**("00","10","11","01"),  lty=**c**(2,3,4,5), col=**c**(2,3,4,5)) |

Figure 7.1 plots the probalility of choosing *coke* with respect to the price ratio for the four possible combinations of the indicator variables *disp*\_*coke* and *disp*\_*pepsi*. In addition, the graph shows the observation points, all located at a height of either 0 or 1. In the legend, the first digit is 0 if *coke* is not displayed and 1 otherwise; the second digit does the same for *pepsi*. Thus, the highest probability of choosing *coke* for any given *pratio* happens when *coke* is displayed and *pepsi* is not (the line marked 10).

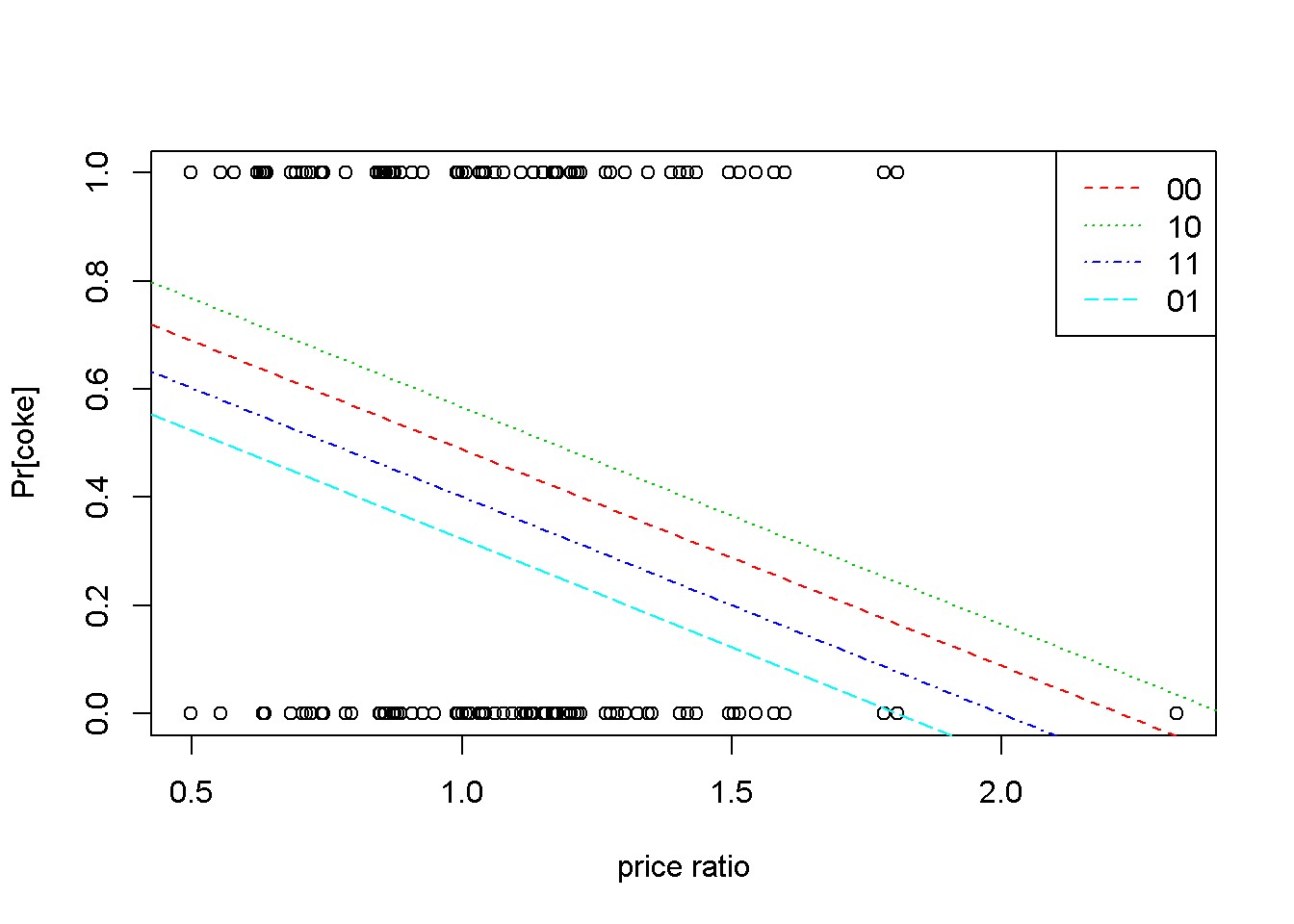


Figure 7.1: Linear probability: the ’coke’ example

## 7.6 Treatment Effects

*Treatment effects* models aim at measuring the differences between a *treatment* group and a *control* group. The **difference estimator** is the simplest such a model and consists in constructing an indicator variable to distinguish between the two groups. Consider, for example, Equation 7.4, where *di* = 1 if observation *i* belongs to the treatment group and *di* = 0 otherwise.

*yi* = *β*1 + *β*2*di* + *ei* (7.4)

Thus, the expected value of the response is *β*1 for the control group and *β*1 + *β*2 for the treatment group. Put otherwise, the coefficient on the dummy variable represents the difference between the treatment and the control group. Moreover, it turns out that *b*2, the estimator of *β*2 is equal to the difference between the sample averages of the two groups:

*b*2 = ¯*y*1 *− y*¯0 (7.5)

The dataset *star* contains information on students from kindergarten to the third grade and was built to identify the determinants of student performance. Let us use this dataset for an example of a difference estimator, as well as an example of selecting just part of the variables and observations in a dataset.

The next piece of code presents a few new elements, that are used to select only a set of all the variables in the dataset, then to split the dataset in two parts: one for small classes, and one for regular classes. Lets look at these elements in the order in which they appear in the code. The function attach(datasetname) allows subsequent use of variable names without specifying from which database they come. While in general doing this is not advisable, I used it to simplify the next lines of code, which lists the variables I want to extract and assigns them to variable vars. As soon as I am done with splitting my dataset, I detach() the dataset to avoid subsequent confusion.

The line starregular is the one that actually retrieves the wanted variables and observations from the dataset. In essence, it picks from the dataset star the lines for whichsmall==0 (see, no need for star$small as long as star is still attached).

The same about the variable starsmall. Another novelty in the following code fragment is the use of the *R* function stargazer() to vizualize a summary table instead of the usual function summary; stargazer shows a more familiar version of the summary statistics table.

|  |
| --- |
| *# Project STAR, an application of the simple difference estimator* **data**("star", package="PoEdata") **attach**(star)  vars <- **c**("totalscore","small","tchexper","boy",  "freelunch","white\_asian","tchwhite","tchmasters",  "schurban","schrural")  starregular <- star[**which**(small==0),vars] starsmall <- star[**which**(small==1),vars] **detach**(star)  **stargazer**(starregular, type=.stargazertype, header=FALSE, title="Dataset 'star' for regular classes") |

**stargazer**(starsmall, type=.stargazertype, header=FALSE, title="Dataset 'star' for small classes")

The two tables titled “Dataset’star’...” display summary statistics for a number of variables in dataset ‘star’. The difference in the means of totalscore is equal to

13*.*740553, which should be equal to the coefficient of the dummy variable *small* in

Equation 7.6. (Note: the results for this dataset seem to be slightly different from

*PoE*4 because the datasets used are of different sizes.)

*totalscore* = *β*1 + *β*2*small* + *e* (7.6)

Table 7.8: Dataset ’star’ for regular classes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Statistic | N | Mean | St. Dev. | Min | Max |
| Totalscore | 4,048 | 918.201 | 72.214 | 635 | 1,253 |
| Small | 4,048 | 0.000 | 0.000 | 0 | 0 |
| Tchexper | 4,028 | 9.441 | 5.779 | 0 | 27 |
| Boy | 4,048 | 0.513 | 0.500 | 0 | 1 |
| Freelunch | 4,048 | 0.486 | 0.500 | 0 | 1 |
| white\_asian | 4,048 | 0.673 | 0.469 | 0 | 1 |
| Tchwhite | 4,048 | 0.824 | 0.381 | 0 | 1 |
| Tchmasters | 4,048 | 0.366 | 0.482 | 0 | 1 |
| Schurban | 4,048 | 0.316 | 0.465 | 0 | 1 |
| Schrural | 4,048 | 0.475 | 0.499 | 0 | 1 |
| Table | 7.9: Da | taset ’star’ | for small cl | asses |  |
| Statistic | N | Mean | St. Dev. | Min | Max |
| Totalscore | 1,738 | 931.942 | 76.359 | 747 | 1,253 |
| Small | 1,738 | 1.000 | 0.000 | 1 | 1 |
| Tchexper | 1,738 | 8.995 | 5.732 | 0 | 27 |
| Boy | 1,738 | 0.515 | 0.500 | 0 | 1 |
| Freelunch | 1,738 | 0.472 | 0.499 | 0 | 1 |
| white\_asian | 1,738 | 0.685 | 0.465 | 0 | 1 |
| Tchwhite | 1,738 | 0.862 | 0.344 | 0 | 1 |
| Tchmasters | 1,738 | 0.318 | 0.466 | 0 | 1 |
| Schurban | 1,738 | 0.306 | 0.461 | 0 | 1 |
| Schrural | 1,738 | 0.463 | 0.499 | 0 | 1 |

mod3 <- **lm**(totalscore~small, data=star) b2 <- **coef**(mod3)[["small"]]

The difference estimated based on Equation 7.6 is *b*2 = 13*.*740553, which coincides with the difference in means we calculated above.

If the students were randomly assigned to small or regular classes and the number of observations is large there is no need for additional regressors (there is no omitted variable bias if other regressors are not correlated with the variable small). In some instances, however, including other regressors may improve the difference estimator. Here is a model that includes a ‘teacher experience’ variable and some school characteristics (the students were randomized within schools, but schools were not randomized).

school <- **as.factor**(star$schid)*#creates dummies for schools* mod4 <- **lm**(totalscore~small+tchexper, data=star) mod5 <- **lm**(totalscore~small+tchexper+school, data=star)

b2n <- **coef**(mod4)[["small"]] b2s <- **coef**(mod5)[["small"]] **anova**(mod4, mod5)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Res.Df | RSS | Df | Sum of Sq | F | Pr(>F) |
| 5763 | 30777461 | NA | NA | NA | NA |
| 5685 | 24072033 | 78 | 6705428 | 20.3025 | 0 |

By the introduction of school *fixed effects* the difference estimate have increased from

14*.*306725 to 15*.*320602. The anova() function, which tests the equivalence of the two regressions yields a very low *p*-value indicating that the difference between the two models is significant.

I have mentioned already, in the context of collinearity that a way to check if there is a relationship between an independent variable and the others is to regress that variable on the others and check the overall significance of the regression. The same method allows checking if the assignments to the treated and control groups are random. If we regress small on the other regressors and the assignment was random we should find no significant relationship.

mod6 <- **lm**(small~boy+white\_asian+tchexper+freelunch, data=star) **kable**(**tidy**(mod6),

caption="Checking random assignment in the 'star' dataset")

fstat <- **glance**(mod6)$statistic pf <- **glance**(mod6)$p.value

The *F*-statistic of the model in Table 7.10 is *F* = 2*.*322664 and its corresponding *p*-value 0*.*054362, which shows that the model is overall insignificant at the 5% level

*7.7. THE DIFFERENCE-IN-DIFFERENCES ESTIMATOR*

Table 7.10: Checking random assignment in the ’star’ dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Term | estimate | std.error | statistic | p.value |
| (Intercept) | 0.325167 | 0.018835 | 17.264354 | 0.000000 |
| Boy | -0.000254 | 0.012098 | -0.021014 | 0.983235 |
| white\_asian | 0.012360 | 0.014489 | 0.853110 | 0.393634 |
| Tchexper | -0.002979 | 0.001055 | -2.825242 | 0.004741 |
| Freelunch | -0.008800 | 0.013526 | -0.650570 | 0.515350 |

or lower. The coefficients of the independent variables are insignificant or extremely small (the coefficient on *tchexper* is statistically significant, but its marginal effect is a change in the probability that *small* = 1 of about 0.3 percent). Again, these results provide fair evidence that the students’ assignment to small or regular classes was random.

Lab 4 -The Difference-in-Differences Estimator

In many economic problems, which do not benefit from the luxury of random samples, the selection into one of the two groups is by choice, thus introducing a selection bias. The difference estimator is biased to the extent to which selection bias is present, therefore it is inadequate when selection bias may be present. The more complex **difference-in-differences** estimator is more appropriate in such cases. While the simple *difference* estimator assumes that before the treatment all units are identical or, at least, that the assignment to the treatment and control group is random, the *difference-in-differences* estimator takes into account any initial heterogeneity between the two groups.

The two ‘differences’ in the *diference-in-differences* estimator are: (i) the difference in the means of the *treatment* and *control* groups in the response variable **after** the treatment, and (ii) the difference in the means of the *treatment* and *control* groups in the response variable **before** the treatment. The ‘difference’ is between *after* and *before*.

Let us denote four averages of the response, as follows:

* *y*¯*T,A* Treatment, After
* *y*¯*C,A* Control, After
* *y*¯*T,B* Treatment, Before
* *y*¯*C,B* Control, Before

The *difference-in-differences* estimator *δ*ˆ is defined as in Equation 7.7.

*δ*ˆ = (¯*yT,A − y*¯*C,A*) *−* (¯*yT,B − y*¯*C,B*) (7.7)

Instead of manually calculating the four means and their difference-in-differences, it is possible to estimate the *difference-in-differences* estimator and its statistical properties by running a regression that includes indicator variables for *treatment* and *after* and their interaction term. The advantage of a regression over simply using Equation 7.7 is that the regression allows taking into account other factors that might influence the treatment effect. The simplest *difference-in-differences* regression model is presented in Equation 7.8, where *yit* is the response for unit *i* in period *t*. In the typical *difference-in-differences* model there are only two periods, *before* and *after*.

*yit* = *β*1 + *β*2*T* + *β*3*A* + *δT × A* + *eit* (7.8)

With a litle algebra it can be seen that the coefficinet *δ* on the interaction term in

Equation 7.8 is exactly the *difference-in-differences* estimator defined in Equation

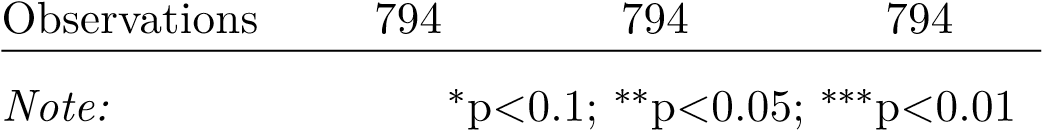
7.7. The following example calculates this estimator for the dataset *njmin*3, where the response is *fte*, the full-time equivalent employment, *d* is the *after* dummy, with *d* = 1 for the *after* period and *d* = 0 for the *before* period, and *nj* is the dummy that marks the treatment group (*nji* = 1 if unit *i* is in New Jersey where the minimum wage law has been changed, and *nji* = 0 if unit *i* in Pennsylvania, where the minimum wage law has not changed). In other words, units (fast-food restaurants) located in New Jersey form the treatment group, and units located in Pennsylvania form the control group.

|  |
| --- |
| **data**("njmin3", package="PoEdata") mod1 <- **lm**(fte~nj\*d, data=njmin3) mod2 <- **lm**(fte~nj\*d+ kfc+roys+wendys+co\_owned, data=njmin3)  mod3 <- **lm**(fte~nj\*d+ kfc+roys+wendys+co\_owned+ southj+centralj+pa1, data=njmin3)  **stargazer**(mod1,mod2,mod3, type=.stargazertype,  title="Difference in Differences example", header=FALSE, keep.stat="n",digits=2  *# single.row=TRUE, intercept.bottom=FALSE*  ) |

*# t-ratio for delta, the D-in-D estimator:* tdelta <- **summary**(mod1)$coefficients[4,3]

*7.7. THE DIFFERENCE-IN-DIFFERENCES ESTIMATOR*

Table 7.11: Difference in Differences example

*Dependent variable:*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | fte |  |
|  | (1) | (2) | (3) |
| Nj | *−*2.89*∗∗* | *−*2.38*∗∗* | *−*0.91 |
|  | (1.19) | (1.08) | (1.27) |
| D | *−*2.17 | *−*2.22 | *−*2.21 |
|  | (1.52) | (1.37) | (1.35) |
| Kfc |  | *−*10.45*∗∗∗* | *−*10.06*∗∗∗* |
|  |  | (0.85) | (0.84) |
| Roys |  | *−*1.62*∗* | *−*1.69*∗∗* |
|  |  | (0.86) | (0.86) |
| Wendys |  | *−*1.06 | *−*1.06 |
|  |  | (0.93) | (0.92) |
| co\_owned |  | *−*1.17 | *−*0.72 |
|  |  | (0.72) | (0.72) |
| Southj |  |  | *−*3.70*∗∗∗*  (0.78) |
| Centralj |  |  | 0.01  (0.90) |
| pa1 |  |  | 0.92  (1.38) |
| nj:d | 2.75 | 2.85*∗* | 2.81*∗* |
|  | (1.69) | (1.52) | (1.50) |
| Constant | 23.33*∗∗∗* | 25.95*∗∗∗* | 25.32*∗∗∗* |
| (1.07) (1.04) (1.21) | | | |

The coefficient on the term *nj* : *d* in the Table titled “Difference-in-Differences example” is *δ*, our difference-in-differences estimator. If we want to test the null hypothesis *H*0 : *δ ≥* 0, the rejection region is at the left tail; since the calculated *t*, which is equal to 1*.*630888 is positive, we cannot reject the null hypothesis. In other words, there is no evidence that an increased minimum wage reduces employment at fast-food restaurants.

Figure 7.2 displays the change of *fte* from the period before (*d* = 0) to the period after the change in minimum wage (*d* = 1) for both the treatment and the control groups. The line labeled “counterfactual” shows how the treatment group would have changed in the absence of the treatment, assuming its change would mirror the change in the control group. The graph is plotted using Equation 7.8.

|  |
| --- |
| b1 <- **coef**(mod1)[[1]] b2 <- **coef**(mod1)[["nj"]] b3 <- **coef**(mod1)[["d"]] delta <- **coef**(mod1)[["nj:d"]]  C <- b1+b2+b3+delta  E <- b1+b3  B <- b1+b2  A <- b1  D <- E+(B-A)  *# This creates an empty plot:*  **plot**(1, type="n", xlab="period", ylab="fte", xaxt="n", xlim=**c**(-0.01, 1.01), ylim=**c**(18, 24))  **segments**(x0=0, y0=A, x1=1, y1=E, lty=1, col=2)*#control* **segments**(x0=0, y0=B, x1=1, y1=C, lty=3, col=3)*#treated* **segments**(x0=0, y0=B, x1=1, y1=D, *#counterfactual* lty=4, col=4)  **legend**("topright", legend=**c**("control", "treated",  "counterfactual"), lty=**c**(1,3,4), col=**c**(2,3,4)) **axis**(side=1, at=**c**(0,1), labels=NULL) |

Lab 5 -- Using Panel Data

The *difference-in-differences* method does not require that the same units be observed both periods, since it works with averages before, and after. If we observe a number of units within the same time period, we construct a *cross-section*; if we construct different cross-sections in different periods we obtain a data structure named *repeated cross-sections*. Sometimes, however, we have information about *the same* units over several periods. A data structure in which the same (cross-sectional) units are observed over two or more periods is called a **panel data** and contains more

*7.8. USING PANEL DATA*

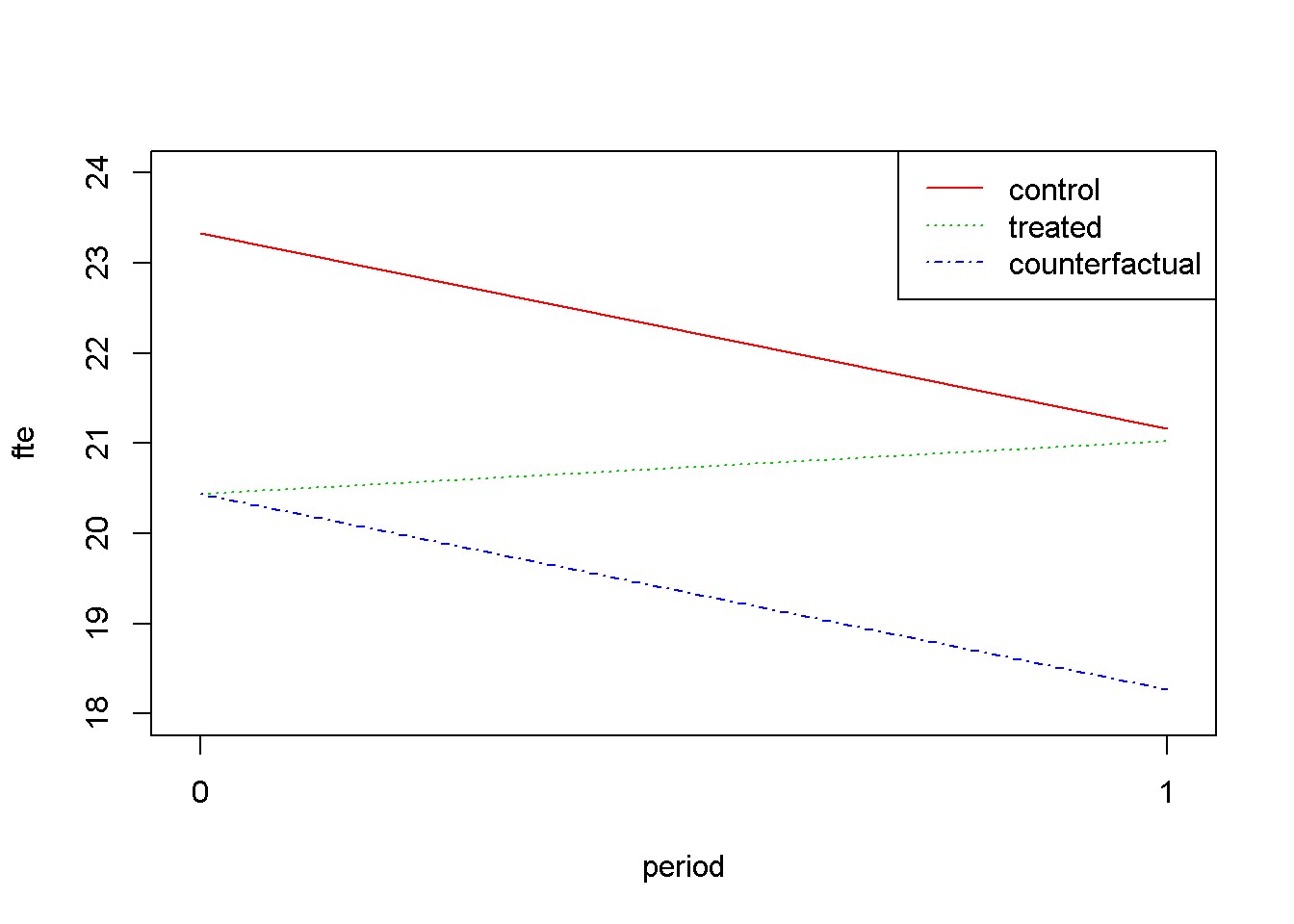


Figure 7.2: Difference-in-Differences for ’njmin3’

information than a repeated cross-section. Let us re-consider the simplest *njmin*3 equation (Equation 7.9), with the unit and time subscripts reflecting the panel data structure of the dataset. The time-invariant term *ci* has been added to reflect unobserved, individual-specific attributes.

*fteit* = *β*1 + *β*2*nji* + *β*3*dt* + *δ*(*nji × dt*) + *ci* + *eit* (7.9)

In the dataset *njmin*3, some restaurants, belonging to either group have been observed both periods. If we restrict the dataset to only those restaurants we obtain a short (two period) panel data. Let us re-calculate the difference-in-differences estimator using this panel data. To do so, we notice that, if we write Equation 7.9 twice, once for each period and subtract the *before* from the *after* equation we obtain Equation 7.10, where the response, *dfte*, is the after- minus- before difference in employment and the only regressor that remains is the *treatment* dummy, *nj*, whose coefficient is *δ*, the very *difference- in- differences* estimator we are trying to estimate.

*dftei* = *α* + *δnji* + *ui* (7.10)

mod3 <- **lm**(demp~nj, data=njmin3) **kable**(**tidy**(**summary**(mod3)),

Table 7.12: Difference in differences with panel data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -2.28333 | 0.731258 | -3.12247 | 0.001861 |
| nj | 2.75000 | 0.815186 | 3.37346 | 0.000780 |

caption="Difference in differences with panel data")

(smod3 <- **summary**(mod3))

##

## Call:

## lm(formula = demp ~ nj, data = njmin3)

##

## Residuals:

## Min 1Q Median 3Q Max

## -39.22 -3.97 0.53 4.53 33.53

##

## Coefficients:

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

## (Intercept) -2.283 0.731 -3.12 0.00186 \*\*

## nj 2.750 0.815 3.37 0.00078 \*\*\*

## ---

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

##

## Residual standard error: 8.96 on 766 degrees of freedom

## (52 observations deleted due to missingness)

## Multiple R-squared: 0.0146, Adjusted R-squared: 0.0134

## F-statistic: 11.4 on 1 and 766 DF, p-value: 0.00078

*R*2 = 0*.*014639*, F* = 11*.*380248*, p* = 0*.*00078

Table 7.12 shows a value of the estimated *difference-in-differences* coefficient very close to the one we estimated before. Its *t*-statistic is still positive, indicating that the null hypothesis *H*0: “an increse in minimum wage increases employment” cannot be rejected.

*7.9. R PRACTICUM*

## 7.9 R Practicum

### 7.9.1 Extracting Various Information

Here is a reminder on how to extract various results after fitting a linear model. The function names(lm.object) returns a list of the names of different items contained in the object. Suppose we run an lm() model and name it mod5. Then, mod5$name returns the item identified by the name. Like about everything in *R*, there are many ways to extract values from an lm() object. I will present three objects that contain about everything we will need. These are the lm() object itself, summary(lm()), and the function glance(lm()) in package broom. The next code shows the name lists for all three objects and a few examples of extracting various statistics.

**library**(broom)

mod5 <- mod5 <- **lm**(wage~educ+black\*female, data=cps4\_small)

smod5 <- **summary**(mod5)

gmod5 <- **glance**(mod5) *#from package 'broom'* **names**(mod5)

## [1] "coefficients" "residuals" "effects" "rank"

## [5] "fitted.values" "assign" "qr" "df.residual"

## [9] "xlevels" "call" "terms" "model"

|  |  |  |
| --- | --- | --- |
| **names**(smod5) | |  |
| ## [1] "call" "terms" "residuals" | | "coefficients" |
| ## [5] "aliased" "sigma" "df"  ## [9] "adj.r.squared" "fstatistic" "cov.unscaled" | | "r.squared" |
| **names**(gmod5) |  |  |
| ## [1] "r.squared" | "adj.r.squared" "sigma" | "statistic" |
| ## [5] "p.value" | "df" "logLik" | "AIC" |
| ## [9] "BIC" | "deviance" "df.residual" |  |
| *# Examples:*  **head**(mod5$fitted.values) | |  |
| ## 1 2 3 4 5 6  ## 23.0605 19.5635 23.6760 18.5949 19.5635 19.5635 | |  |
| **head**(mod5$residuals)*#head of residual vector* | |  |

## 1 2 3 4 5 6 ## -4.360484 -8.063529 -8.636014 7.355080 4.466471 0.436471 smod5$r.squared

## [1] 0.208858

smod5$fstatistic *# F-stat. and its degrees of freedom*

## value numdf dendf

## 65.6688 4.0000 995.0000

gmod5$statistic *#the F-statistic of the model*

## [1] 65.6688

mod5$df.residual

## [1] 995

For some often used statistics, such as coefficients and their statistics, fitted values, or residuals there are specialized functions to extract them from regression results.

|  |
| --- |
| N <- **nobs**(mod5)  yhat <- **fitted**(mod5) *# fitted values* ehat <- **resid**(mod5) *# estimated residuals*  allcoeffs <- **coef**(mod5) *# only coefficients, no statistics* **coef**(mod5)[[2]] *#or:* |

## [1] 2.07039

**coef**(mod5)[["educ"]]

## [1] 2.07039

**coeftest**(mod5)*# all coefficients and their statistics*

##

## t test of coefficients:

##

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

## (Intercept) -5.2812 1.9005 -2.779 0.00556 \*\*

## educ 2.0704 0.1349 15.350 < 2e-16 \*\*\*

## black -4.1691 1.7747 -2.349 0.01901 \*

## female -4.7846 0.7734 -6.186 8.98e-10 \*\*\* ## black:female 3.8443 2.3277 1.652 0.09894 .

## ---

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

*# The 'tidy()' function is from the package 'broom':*

tabl <- **tidy**(mod5) *#this gives the same result as coeftest()*

*7.9. R PRACTICUM*

Table 7.13: Example of using the function ’tidy’

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| term | estimate | std.error | statistic | p.value |
| (Intercept) | -5.28116 | 1.900468 | -2.77887 | 0.005557 |
| educ | 2.07039 | 0.134878 | 15.35009 | 0.000000 |
| black | -4.16908 | 1.774714 | -2.34916 | 0.019011 |
| female | -4.78461 | 0.773414 | -6.18635 | 0.000000 |
| black:female | 3.84429 | 2.327653 | 1.65158 | 0.098937 |

**kable**(tabl, caption=" Example of using the function 'tidy'")

### 7.9.2 ggplot2, An Excellent Data Visualising Tool

The function ggplot in package ggplot2 is a very flexible plotting tool. For instance, it can assign different colours to different levels of an indicator variable. The following example uses the file *utown* to plot *price* against *sqft* using different colours for houses with swimming pool or for houses close to university.

|  |
| --- |
| **library**(ggplot2) **data**("utown", package="PoEdata") fpool <- **as.factor**(utown$pool) futown <- **as.factor**(utown$utown) **ggplot**(data = utown) +  **geom\_point**(mapping = **aes**(x = sqft, y = price, color = fpool, shape=fpool))  **ggplot**(data = utown) +  **geom\_point**(mapping = **aes**(x = sqft, y = price,  color = futown, shape=futown)) |

A brilliant, yet concise presentation of the ggplot() system can be found in (Grolemund and Wickham 2016), which I strongly recommend.

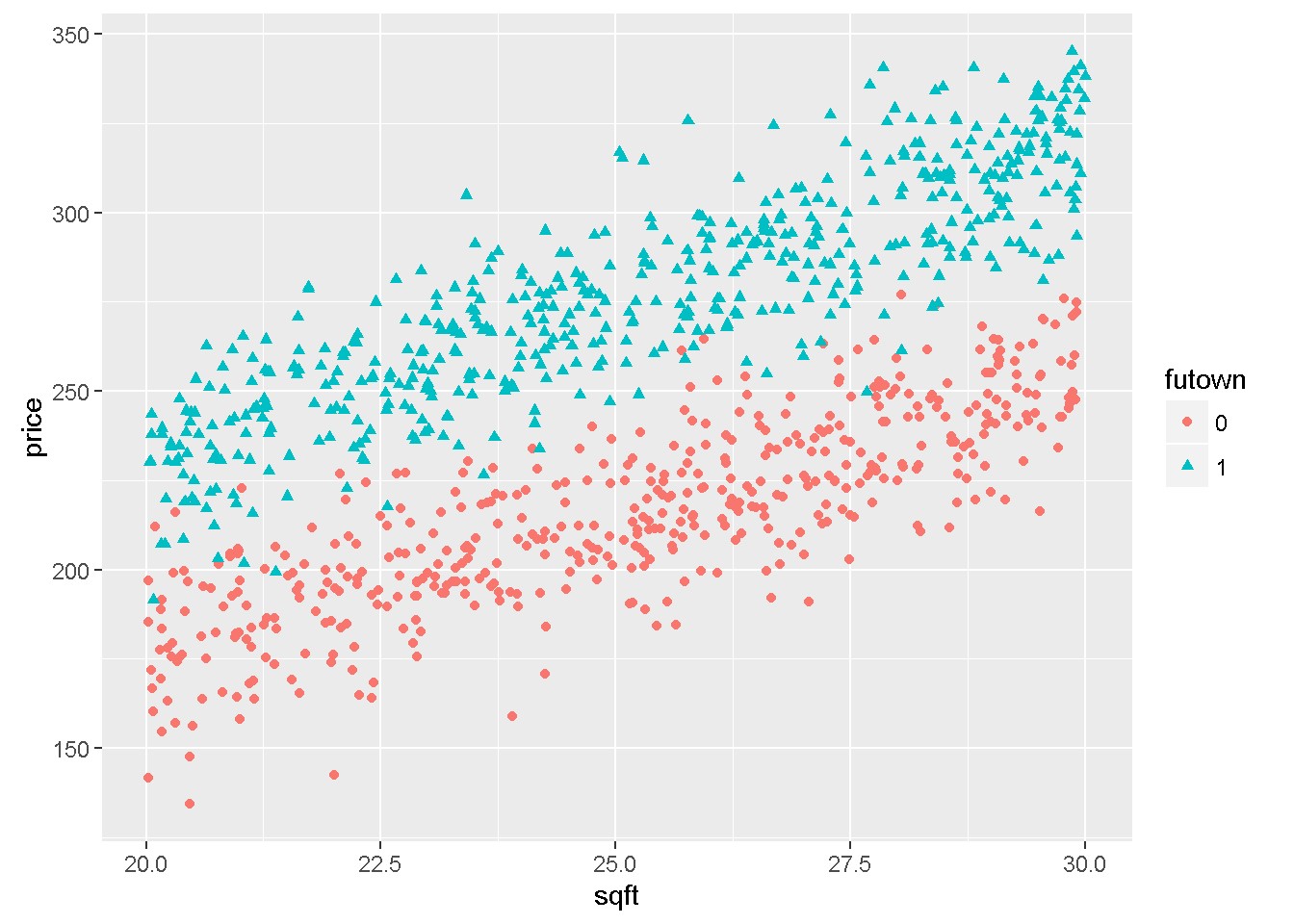


Figure 7.3: Graphs of dataset ’utown’ using the ’ggplot’ function

Lab 6 - Heteroskedasticity

**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)

Reference for the package sandwich (Lumley and Zeileis 2015).

One of the assumptions of the Gauss-Markov theorem is **homoskedasticity**, which requires that all observations of the response (dependent) variable come from distributions with the same variance *σ*2. In many economic applications, however, the spread of *y* tends to depend on one or more of the regressors *x*. For example, in the *food* simple regression model (Equation 8.1) expenditure on food stays closer to its mean (regression line) at lower incomes and to be more spread about its mean at higher incomes. Think just that people have more choices at higher income whether to spend their extra income on food or something else.

*food*\_*expi* = *β*1 + *β*2*incomei* + *ei* (8.1)

In the presence of heteroskedasticity, the coefficient estimators are still unbiased, but their variance is incorrectly calculated by the usual OLS method, which makes confidence intervals and hypothesis testing incorrect as well. Thus, new methods need to be applied to correct the variances.

121

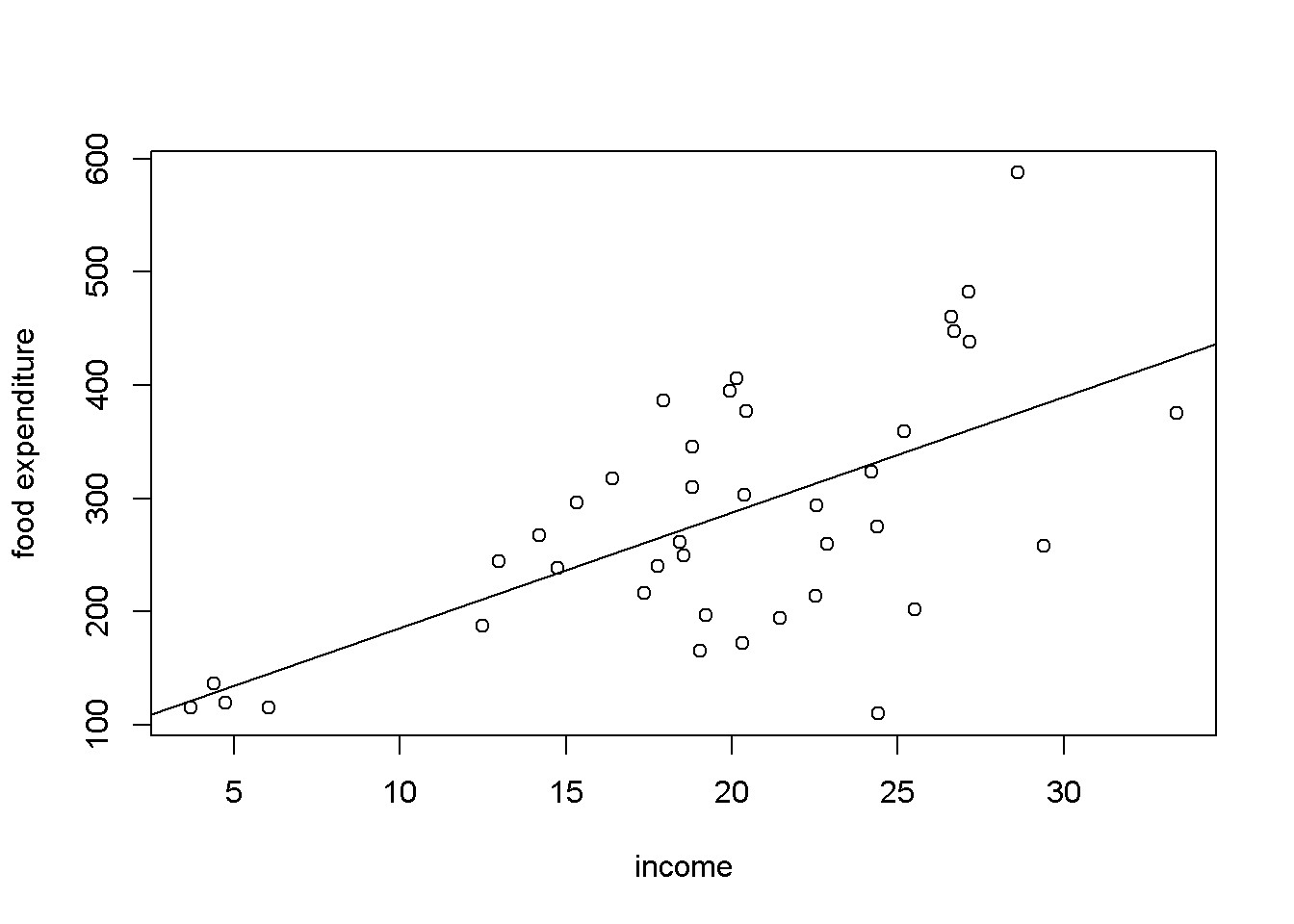


Figure 8.1: Heteroskedasticity in the ’food’ data

## 8.1 Spotting Heteroskedasticity in Scatter Plots

When the variance of *y*, or of *e*, which is the same thing, is not constant, we say that the response or the residuals are **heteroskedastic**. Figure 8.1 shows, again, a scatter diagram of the *food* dataset with the regression line to show how the observations tend to be more spread at higher income.

**data**("food",package="PoEdata") mod1 <- **lm**(food\_exp~income, data=food) **plot**(food$income,food$food\_exp, type="p", xlab="income", ylab="food expenditure")

**abline**(mod1)

Another useful method to visualize possible heteroskedasticity is to plot the residuals against the regressors suspected of creating heteroskedasticity, or, more generally, against the fitted values of the regression. Figure 8.2 shows both these options for the simple *food\_exp* model.

|  |
| --- |
| res <- **residuals**(mod1) yhat <- **fitted**(mod1)  **plot**(food$income,res, xlab="income", ylab="residuals") **plot**(yhat,res, xlab="fitted values", ylab="residuals") |

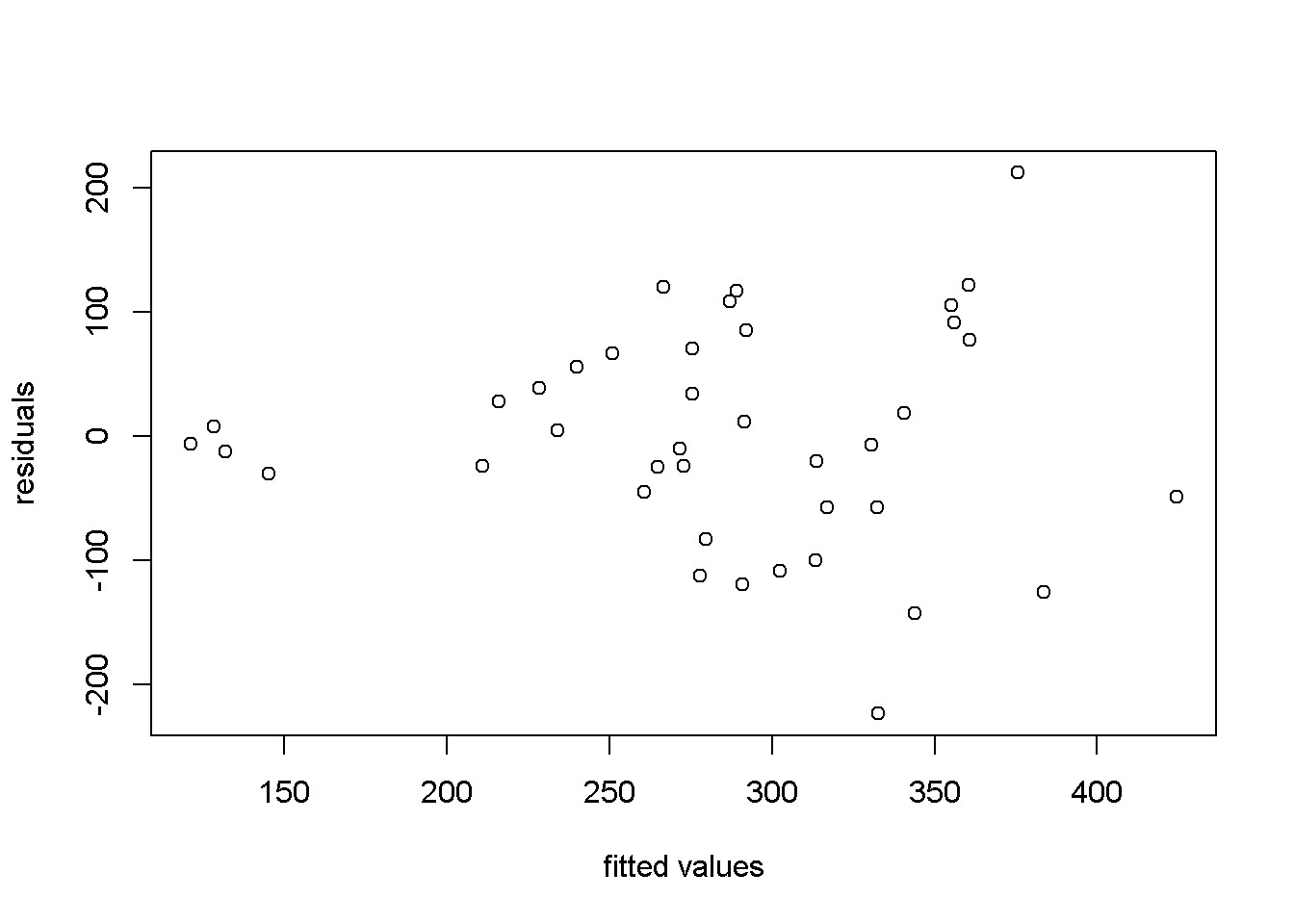
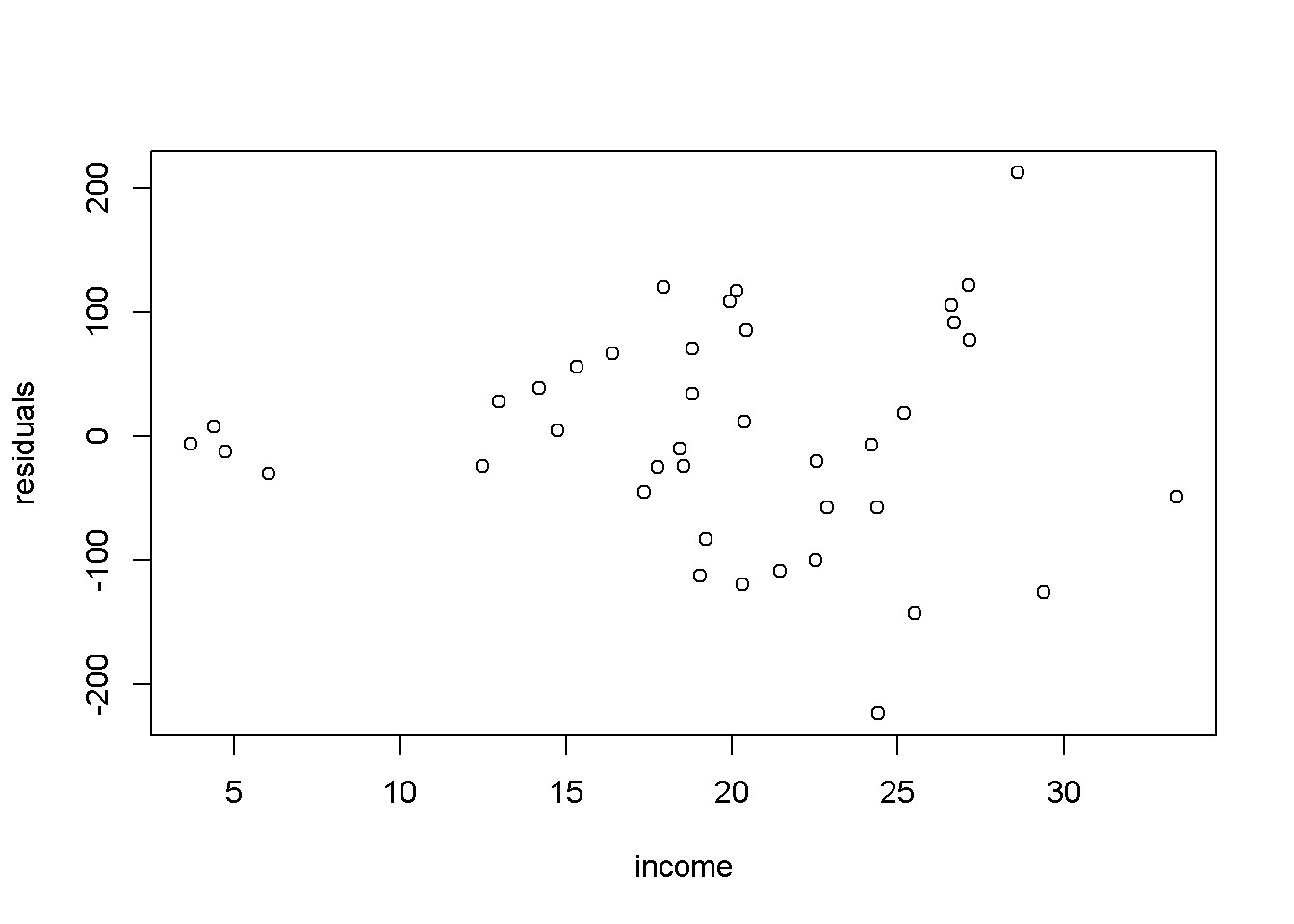


Figure 8.2: Residual plots in the ’food’ model

## 8.2 Heteroskedasticity Tests

Suppose the regression model we want to test for heteroskedasticity is the one in Equation 8.2.

*yi* = *β*1 + *β*2*xi*2 + *...* + *βKxiK* + *ei* (8.2)

The test we are construction assumes that the variance of the errors is a function *h* of a number of regressors *zs*, which may or may not be present in the initial regression model that we want to test. Equation 8.3 shows the general form of the variance function.

|  |  |
| --- | --- |
| *var*(*yi*) = *E*(*e*2*i* ) = *h*(*α*1 + *α*2*zi*2 + *...* + *αSziS*) | (8.3) |

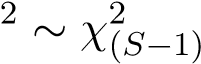
The variance *var*(*yi*) is constant only if all the coefficients of the regressors *z* in Equation 8.3 are zero, which provides the null hypothesis of our heteroskedasticity test shown in Equation 8.4.

*H*0 : *α*2 = *α*3 = *...αS* = 0 (8.4)

Since we do not know the true variances of the response variable *yi*, all we can do is to estimate the residuals from the initial model in Equation 8.2 and replace *e*2*i* in Equation 8.3 with the estimated residuals. For a linear function *h*(), the test equation is, finally, Equation 8.5.

*e*ˆ2*i* = *α*1 + *α*2*zi*2 + *...* + *αSziS* + *νi* (8.5)

Te relevant test statistic is *χ*2, given by Equation 8.6, where *R*2 is the one resulted from Equation 8.5.

*χ*2 = *N × R* (8.6)

The **Breusch-Pagan** heteroskedasticiy test uses the method we have just described, where the regressors *zs* are the variables *xk* in the initial model. Let us apply this test to the *food* model. The function to determine a critical value of the *χ*2 distribution for a significance level *α* and *S −* 1 degrees of freedom is qchisq(1-alpha, S-1).

|  |
| --- |
| alpha <- 0.05  mod1 <- **lm**(food\_exp~income, data=food) ressq <- **resid**(mod1)^2 *#The test equation:* modres <- **lm**(ressq~income, data=food) N <- **nobs**(modres) gmodres <- **glance**(modres)  S <- gmodres$df *#Number of Betas in model #Chi-square is always a right-tail test* chisqcr <- **qchisq**(1-alpha, S-1) Rsqres <- gmodres$r.squared chisq <- N\*Rsqres pval <- 1-**pchisq**(chisq,S-1) |

Our test yields a value of the test statistic *χ*2 of 7*.*38, which is to be compared to the critical *χ*2*cr* having *S −* 1 = 1 degrees of freedom and *α* = 0*.*05. This critical value is *χ*2*cr* = 3*.*84. Since the calculated *χ*2 exceeds the critical value, we reject the null hypothesis of homoskedasticity, which means there is heteroskedasticity in our data and model. Alternatively, we can find the *p*-value corresponding to the calculated *χ*2, *p* = 0*.*007.

Let us now do the same test, but using a **White** version of the residuals equation, in its quadratic form.

|  |
| --- |
| modres <- **lm**(ressq~income+**I**(income^2), data=food)  gmodres <- **glance**(modres)  Rsq <- gmodres$r.squared  S <- gmodres$df *#Number of Betas in model* chisq <- N\*Rsq  pval <- 1-**pchisq**(chisq, S-1) |

The calculated *p*-value in this version is *p* = 0*.*023, which also implies rejection of the null hypothesis of homoskedasticity.

Table 8.1: Breusch-Pagan heteroskedasticity test

|  |  |  |  |
| --- | --- | --- | --- |
| statistic | p.value | parameter | method |
| 7.38442 | 0.006579 | 1 | studentized Breusch-Pagan test |

The function bptest() in package lmtest does (the robust version of) the Breusch-

Pagan test in *R*. The following code applies this function to the basic *food* equation, showing the results in Table 8.1, where ‘statistic’ is the calculated *χ*2.

mod1 <- **lm**(food\_exp~income, data=food) **kable**(**tidy**(**bptest**(mod1)), caption="Breusch-Pagan heteroskedasticity test")

The **Goldfeld-Quandt** heteroskedasticity test is useful when the regression model to be tested includes an indicator variable among its regressors. The test compares the variance of one group of the indicator variable (say group 1) to the variance of the benchmark group (say group 0), as the null hypothesis in Equation8.7 shows.

*H*0 : *σ*ˆ12 = *σ*ˆ02*, HA* : *σ*ˆ12 *6*= *σ*ˆ02 (8.7)

The test statistic when the null hyppthesis is true, given in Equation 8.8, has an *F* distribution with its two degrees of freedom equal to the degrees of freedom of the two subsamples, respectively *N*1 *− K* and *N*0 *− K*.

*σ*ˆ2

*F* =  (8.8)

*σ*ˆ0

Let us apply this test to a *wage* equation based on the dataset *cps*2, where *metro* is an indicator variable equal to 1 if the individual lives in a metropolitan area and 0 for rural area. I will split the dataset in two based on the indicator variable *metro* and apply the regression model (Equation 8.9) separately to each group.

*wage* = *β*1 + *β*2*educ* + *β*3*exper* + *β*4*metro* + *e* (8.9)

alpha <- 0.05 *#two tail, will take alpha/2* **data**("cps2", package="PoEdata")

*#Create the two groups, m (metro) and r (rural)*

m <- cps2[**which**(cps2$metro==1),] r <- cps2[**which**(cps2$metro==0),] wg1 <- **lm**(wage~educ+exper, data=m) wg0 <- **lm**(wage~educ+exper, data=r)

|  |
| --- |
| df1 <- wg1$df.residual *#Numerator degrees of freedom* df0 <- wg0$df.residual *#Denominatot df* sig1squared <- **glance**(wg1)$sigma^2 sig0squared <- **glance**(wg0)$sigma^2 fstat <- sig1squared/sig0squared  Flc <- **qf**(alpha/2, df1, df0)*#Left (lower) critical F*  Fuc <- **qf**(1-alpha/2, df1, df0) *#Right (upper) critical F* |

The results of these calculations are as follows: calculated *F* statistic *F* = 2*.*09, the lower tail critical value *Flc* = 0*.*81, and the upper tail critical value *Fuc* = 1*.*26. Since the calculated amount is greater than the upper critical value, we reject the hypothesis that the two variances are equal, facing, thus, a heteroskedasticity problem. If one expects the variance in the metropolitan area to be higher and wants to test the (alternative) hypothesis *H**, HA* : *σ*12 *> σ*02, one needs to re-calcuate the critical value for *α* = 0*.*05 as follows:

Fc <- **qf**(1-alpha, df1, df0) *#Right-tail test*

The critical value for the right tail test is *Fc* = 1*.*22, which still implies rejecting the null hypothesis.

The Goldfeld-Quant test can be used even when there is no indicator variable in the model or in the dataset. One can split the dataset in two using an arbitrary rule. Let us apply the method to the basic *food* equation, with the data split in low-income (*li*) and high-income (*hi*) halves. The cutoff point is, in this case, the median income, and the hypothesis to be tested

*H**, HA* : *σhi*2 *> σli*2

|  |
| --- |
| alpha <- 0.05  **data**("food", package="PoEdata") medianincome <- **median**(food$income) li <- food[**which**(food$income<=medianincome),] hi <- food[**which**(food$income>=medianincome),] eqli <- **lm**(food\_exp~income, data=li) eqhi <- **lm**(food\_exp~income, data=hi) dfli <- eqli$df.residual dfhi <- eqhi$df.residual sigsqli <- **glance**(eqli)$sigma^2 sigsqhi <- **glance**(eqhi)$sigma^2  fstat <- sigsqhi/sigsqli *#The larger var in numerator* Fc <- **qf**(1-alpha, dfhi, dfli) pval <- 1-**pf**(fstat, dfhi, dfli) |

Table 8.2: R function ‘gqtest()‘ with the ’food’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| df1 | df2 | statistic | p.value | method |
| 18 | 18 | 3.61476 | 0.004596 | Goldfeld-Quandt test |

The resulting *F* statistic in the *food* example is *F* = 3*.*61, which is greater than the critical value *Fcr* = 2*.*22, rejecting the null hypothesis in favour of the alternative hypothesis that variance is higher at higher incomes. The *p*-value of the test is *p* = 0*.*0046.

In the package lmtest, *R* has a specialized function to perform Goldfeld-Quandt tests, the function gqtest which takes, among other arguments, the formula describing the model to be tested, a break point specifying how the data should be split (percentage of the number of observations), what is the alternative hypothesis (“greater”, “two.sided”, or “less”), how the data should be ordered (order.by=), and data=. Let us apply gqtest() to the *food* example with the same partition as we have just did before.

foodeq <- **lm**(food\_exp~income, data=food) tst <- **gqtest**(foodeq, point=0.5, alternative="greater", order.by=food$income) **kable**(**tidy**(tst),

caption="R function `gqtest()` with the 'food' equation")

Please note that the results from applying gqtest() (Table 8.2 are the same as those we have already calculated.

## 8.3 Heteroskedasticity-Consistent Standard Errors

Since the presence of heteroskedasticity makes the lest-squares standard errors incorrect, there is a need for another method to calculate them. White robust standard errors is such a method.

The *R* function that does this job is hccm(), which is part of the car package and yields a heteroskedasticity-robust coefficient covariance matrix. This matrix can then be used with other functions, such as coeftest() (instead of summary), waldtest() (instead of anova), or linearHypothesis() to perform hypothesis testing. The function hccm() takes several arguments, among which is the model for which we want the robust standard errors and the type of standard errors we wish to calculate. type can be “constant” (the regular homoskedastic errors), “hc0”, “hc1”, “hc2”,

“hc3”, or “hc4”; “hc1” is the default type in some statistical software packages. Let Table 8.3: Regular standard errors in 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 |

Table 8.4: Robust (HC1) standard errors in the ’food’ equation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

us compute robust standard errors for the basic *food* equation and compare them with the regular (incorrect) ones.

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

**kable**(**tidy**(foodeq),caption=

"Regular standard errors in the 'food' equation")

|  |
| --- |
| cov1 <- **hccm**(foodeq, type="hc1") *#needs package 'car'* food.HC1 <- **coeftest**(foodeq, vcov.=cov1) **kable**(**tidy**(food.HC1),caption=  "Robust (HC1) standard errors in the 'food' equation") |

When comparing Tables 8.3 and 8.4, it can be observed that the robust standard errors are smaller and, since the coefficients are the same, the *t*-statistics are higher and the *p*-values are smaller. Lower *p*-values with robust standard errors is, however, the exception rather than the rule.

Next is an example of using robust standard errors when performing a fictitious linear hypothesis test on the basic ‘andy’ model, to test the hypothesis *H*0 : *β*2 + *β*3 = 0

|  |
| --- |
| **data**("andy", package="PoEdata")  andy.eq <- **lm**(sales~price+advert, data=andy) bp <- **bptest**(andy.eq) *#Heteroskedsticity test* b2 <- **coef**(andy.eq)[["price"]] b3 <- **coef**(andy.eq)[["advert"]] H0 <- "price+advert=0"  **kable**(**tidy**(**linearHypothesis**(andy.eq, H0, vcov=**hccm**(andy.eq, type="hc1"))),  caption="Linear hypothesis with robust standard errors") |

Table 8.5: Linear hypothesis with robust standard errors

|  |  |  |  |
| --- | --- | --- | --- |
| res.df | df | statistic | p.value |
| 73 | NA | NA | NA |
| 72 | 1 | 23.387 | 7e-06 |

Table 8.6: Linear hypothesis with regular standard errors

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| res.df | Rss | df | sumsq | statistic | p.value |
| 73 | 2254.71 | NA | NA | NA | NA |
| 72 | 1718.94 | 1 | 535.772 | 22.4415 | 0.000011 |

**kable**(**tidy**(**linearHypothesis**(andy.eq, H0)),

caption="Linear hypothesis with regular standard errors")

This example demonstrates how to introduce robust standards errors in a linearHypothesis function. It also shows that, when heteroskedasticity is not significant (bptst does not reject the homoskedasticity hypothesis) the robust and regular standard errors (and therefore the *F* statistics of the tests) are very similar.

Just for completeness, I should mention that a similar function, with similar uses is the function vcov, which can be found in the package sandwich.