MEM52220 - Applied Econometrics

Nicolas Reigl

2018-08-16

Contents

Chapter 1

Introduction

Welcome to MEM5220 - Applied Econometrics. This handout was originally (and currently) designed for the use with MEM5220 - Applied Econometrics at Tallinn University of Technology. Note that this workbook is still under heavy development!

1.1 Prerequisites

A basic knowledge of the R (**?**) programming language is required.

1.2 Resources

Our primary resource is **?** [1] . For theoretical concepts see **?**.

1.3 Acknowledgements

I thank Kadri Männasoo and Juan Carlos Cuestas for proofreading and their useful comments.

[1] Heiss (2016) builds on the popular Introductory Econometrics by Wooldridge (2016) and demonstrates how to replicate the applications discussed therein using R.

Chapter 2

Linear Regression

The general form in which we specify regression models in R:

```
## response ~ terms
##
## y ~ age + sex            # age + sex main effects
## y ~ age + sex + age:sex  # add second-order interaction
## y ~ age*sex              # second-order interaction +
##                          # all main effects
## y ~ (age + sex + pressure)^2
##                          # age+sex+pressure+age:sex+age:pressure...
## y ~ (age + sex + pressure)^2 - sex:pressure
##                          # all main effects and all 2nd order
##                          # interactions except sex:pressure
## y ~ (age + race)*sex     # age+race+sex+age:sex+race:sex
## y ~ treatment*(age*race + age*sex) # no interact. with race,sex
## sqrt(y) ~ sex*sqrt(age) + race
## # functions, with dummy variables generated if
## # race is an R factor (classification) variable
## y ~ sex + poly(age,2)    # poly generates orthogonal polynomials
## race.sex <- interaction(race,sex)
## y ~ age + race.sex       # for when you want dummy variables for
##                          # all combinations of the factors
```

2.1 Simple Linear Regression

We start off with a simple OLS Regression. We will work with multiple data sources:

- Data from **?** : Introductory Econometrics: A Modern Approch.
- *More datasources in the future*

To load the dataset and necessary functions:

```
# This function 1. checks if the packages are installed. 2. It installs the packages if they were not i
PACKAGES<-c("wooldridge",  # Wooldrige Datasets
            "tidyverse",  # for data manipulation and ggplots
            "broom",  # Tidy regression output
            "ggpubr",  # Multiple ggplots on a page. Note that, the installation of ggpubr will automat
            "ggfortify", # Simple ggplot recipe for lm objects)
```

```
            "plot3D",  #  3D graphs
            "car", # Companion to applied regression
            "knitr", # knit functions
            # "kableExtra", # extended knit functions for objects exported from other packages
            "huxtable", #  Regression tables, broom compatible
            "stargazer", # Regression tables
            "AER", #  Functions, data sets, examples, demos, and vignettes for the book Christian Kleib
            "PoEdata", # R data sets for "Principles of Econometrics" by Hill, Griffiths, and Lim, 4e,
          "summarytools", # Report regression summary tables
          "MCMCpack", # Contains functions to perform Bayesian inference using posterior simulation for
          "sampleSelection", # Two-step and maximum likelihood estimation of Heckman-type sample select
           "scales", # scale helper functions such as percent
            "magrittr") #  pipes
inst<-match(PACKAGES, .packages(all=TRUE))
need<-which(is.na(inst))
if (length(need)>0) install.packages(PACKAGES[need])
lapply(PACKAGES, require, character.only=T)
```

Classic examples of quantities modeled with simple linear regression:

- College GPA   SAT scores $\beta > 0$
- Change in GDP   change in unemployment $\beta < 0$
- House price   number of bedrooms $\beta > 0$
- Species heart weight   species body weight $\beta > 0$
- Fatalities per year   speed limit $\beta < 0$

Notice that these simple linear regressions are simplifications of more complex relationships between the variables in question.

In this exercise we use the dataset *ceosal1*. Let us analyse the dataset first

```
data("ceosal1")
help("ceosal1")
?ceosal1
```

As we see from the R documentation the *ceosal1* dataset contain of a random sample of data reported in the May 6, 1991 issue of Businessweek.

To get a first look at the data you can use the `View()` function inside R Studio.

```
View(ceosal1)
```

We could also take a look at the variable names, the dimension of the data frame, and some sample observations with `str()`.

```
str(ceosal1)
```

```
## 'data.frame':    209 obs. of  12 variables:
##  $ salary  : int  1095 1001 1122 578 1368 1145 1078 1094 1237 833 ...
##  $ pcsalary: int  20 32 9 -9 7 5 10 7 16 5 ...
##  $ sales   : num  27595 9958 6126 16246 21783 ...
##  $ roe     : num  14.1 10.9 23.5 5.9 13.8 ...
##  $ pcroe   : num  106.4 -30.6 -16.3 -25.7 -3 ...
##  $ ros     : int  191 13 14 -21 56 55 62 44 37 37 ...
##  $ indus   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ finance : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ consprod: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ utility : int  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ lsalary : num  7 6.91 7.02 6.36 7.22 ...
##  $ lsales  : num  10.23 9.21 8.72 9.7 9.99 ...
##  - attr(*, "time.stamp")= chr "25 Jun 2011 23:03"
```

As we have seen before in the general R tutorial, there are a number of additional functions to access some of this information directly.

```r
dim(ceosal1)
```

```
## [1] 209  12
```

```r
nrow(ceosal1)
```

```
## [1] 209
```

```r
ncol(ceosal1)
```

```
## [1] 12
```

```r
summary(ceosal1)
```

```
##      salary         pcsalary          sales              roe       
##  Min.   :  223   Min.   :-61.00   Min.   :  175.2   Min.   : 0.50  
##  1st Qu.:  736   1st Qu.: -1.00   1st Qu.: 2210.3   1st Qu.:12.40  
##  Median : 1039   Median :  9.00   Median : 3705.2   Median :15.50  
##  Mean   : 1281   Mean   : 13.28   Mean   : 6923.8   Mean   :17.18  
##  3rd Qu.: 1407   3rd Qu.: 20.00   3rd Qu.: 7177.0   3rd Qu.:20.00  
##  Max.   :14822   Max.   :212.00   Max.   :97649.9   Max.   :56.30  
##      pcroe            ros            indus           finance      
##  Min.   :-98.9   Min.   :-58.0   Min.   :0.0000   Min.   :0.0000  
##  1st Qu.:-21.2   1st Qu.: 21.0   1st Qu.:0.0000   1st Qu.:0.0000  
##  Median : -3.0   Median : 52.0   Median :0.0000   Median :0.0000  
##  Mean   : 10.8   Mean   : 61.8   Mean   :0.3206   Mean   :0.2201  
##  3rd Qu.: 19.5   3rd Qu.: 81.0   3rd Qu.:1.0000   3rd Qu.:0.0000  
##  Max.   :977.0   Max.   :418.0   Max.   :1.0000   Max.   :1.0000  
##     consprod         utility          lsalary          lsales      
##  Min.   :0.0000   Min.   :0.0000   Min.   :5.407   Min.   : 5.166  
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:6.601   1st Qu.: 7.701  
##  Median :0.0000   Median :0.0000   Median :6.946   Median : 8.217  
##  Mean   :0.2871   Mean   :0.1722   Mean   :6.950   Mean   : 8.292  
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:7.249   3rd Qu.: 8.879  
##  Max.   :1.0000   Max.   :1.0000   Max.   :9.604   Max.   :11.489  
```

The interesting task here is to determine how far a high the CEO salary is, for a given return on equity.

Your turn

What sign would be expect of β (the slope)?

A: Without seeing the data **my** prior is that $\beta > 0$.

Note

A simple linear model as assumes that the mean of each y_i conditioned on x_i is a linear function of x_i. But notice that simple linear regressions are simplifications of more complex relationships between the variables in question.

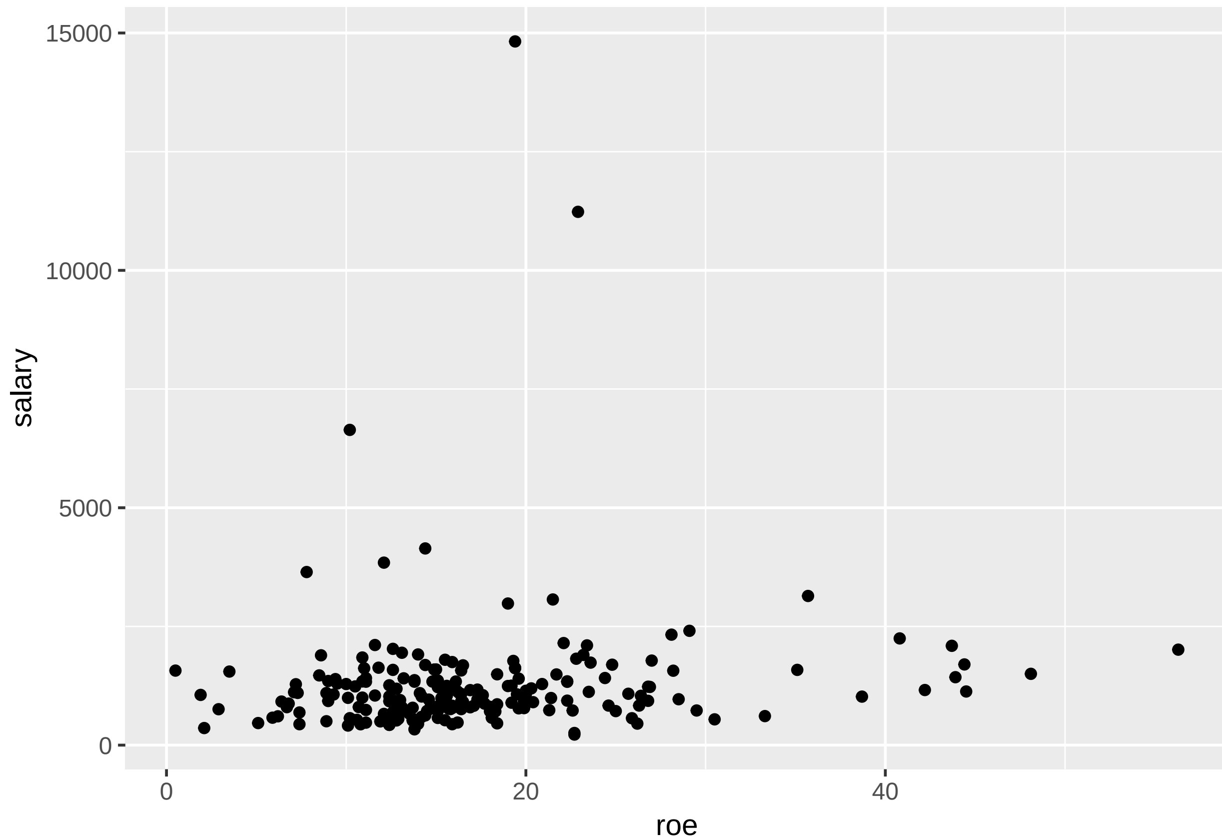


Figure 2.1: Relationship between ROE and Salary

```r
# Use ggplot style
ggplot(ceosal1, aes(x = roe, y = salary)) +
  geom_point()
```

Consider a simple regression model

$salary = \beta_0 + \beta_1 roe + u$

We are concerned with the population parameter β_0 and β_1. The general form of the model.

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2.1}$$

The ordinary least squares (OLS) estimators are

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{2.2}$$

Ingredients for the OLS formulas

```r
attach(ceosal1)
```

```
## The following objects are masked from ceosal1 (pos = 15):
##
##     consprod, finance, indus, lsalary, lsales, pcroe, pcsalary,
##     roe, ros, salary, sales, utility
```

```
cov(roe, salary)
```

```
## [1] 1342.538
```

```
var(roe)
```

```
## [1] 72.56499
```

```
mean(salary)
```

```
## [1] 1281.12
```

Manual calculation of the OLS coefficients

```
b1hat <- cov(roe,salary)/var(roe)
```

```
b0hat <- mean(salary) - b1hat * mean(roe)
```

Or use the `lm()` function

```
lm(salary ~ roe, data=ceosal1)
```

```
##
## Call:
## lm(formula = salary ~ roe, data = ceosal1)
##
## Coefficients:
## (Intercept)          roe
##       963.2         18.5
```

```
lm1_ceosal1 <- lm(salary ~ roe, data=ceosal1)
```

```
unique(ceosal1$roe)
```

```
##   [1] 14.1 10.9 23.5  5.9 13.8 20.0 16.4 16.3 10.5 26.3 25.9 26.8 14.8 22.3
##  [15] 56.3 12.6 20.4  1.9 19.9 15.4 38.7 24.4 15.6 14.4 19.0 16.1 12.1 16.2
##  [29] 18.4 14.2 14.9 12.4 17.1 16.9 18.1 19.3 18.3 13.7 12.7 15.1 16.5 10.2
##  [43] 19.6 12.8 15.9 17.3  8.5 19.5 19.2 28.1 25.0 15.0 20.3 22.7 13.2 10.3
##  [57] 17.7 10.0  6.8 13.1 15.8 15.3  0.5 13.0 11.1  8.9 17.5  9.3  9.5 15.5
##  [71]  8.6 24.6  7.2 11.6 26.4 21.4  9.0  9.4  3.5 22.1 33.3 22.8 20.9  6.7
##  [85]  7.1 11.8 14.0 10.1  6.4 17.6 23.6 35.7 23.2 44.4  2.1 23.4 25.7 27.0
##  [99] 43.7 24.8 26.2 44.5 35.1 11.0 19.4 28.5 43.9 15.7 28.2 42.2 21.5 29.5
## [113] 22.6 22.9  7.8 48.1 18.0 21.7 21.3 26.9 30.5 29.1 40.8 10.8  5.1 12.3
## [127]  7.4  6.2 10.6  2.9 13.5 10.7 11.9 12.9  7.3 14.6 14.5 14.7
```

Plot the linear regression fit the *base* r way.

```
plot(salary~ roe, data = ceosal1,
     xlab = "Return on equity",
     ylab = "Salary",
     main = "Salary vs return on equity",
     pch  = 20,
     cex  = 2,
     col  = "grey")
abline(lm1_ceosal1, lwd = 3, col = "darkorange")
```

Or use ggplot

```
ggplot(ceosal1, aes(x = roe, y = salary)) +
  geom_point() +
```

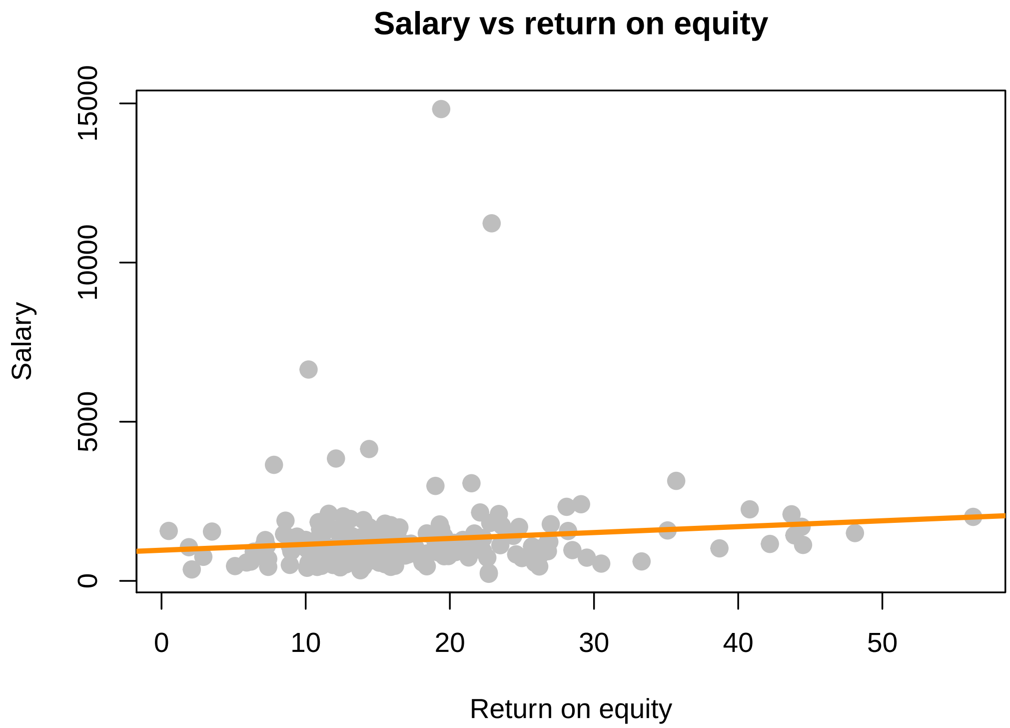


Figure 2.2: OLS regression base Rstyle

```
  stat_smooth(method = "lm", col = "red")
```

Determine the names of the elements of the list using the `names()` command.

```
names(lm1_ceosal1)
```

```
##  [1] "coefficients" "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"       "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

Extract one element, for example the residuals from the list object

```
head(lm1_ceosal1$residuals) # head() just prints out the first 6 residual values
```

```
##         1         2         3         4         5         6
## -129.0581 -163.8543 -275.9692 -494.3483  149.4923 -188.2151
```

Another way to access stored information in *lm1_ceosal1* are the `coef()`, `resid()`, and `fitted()` functions. These return the coefficients, residuals, and fitted values, respectively.

```
coef(lm1_ceosal1)
```

```
## (Intercept)         roe
##   963.19134    18.50119
```

The function `summary()` is useful in many situations. We see that when it is called on our model, it returns a good deal of information.

```
summary(lm1_ceosal1)
```

```
##
## Call:
```

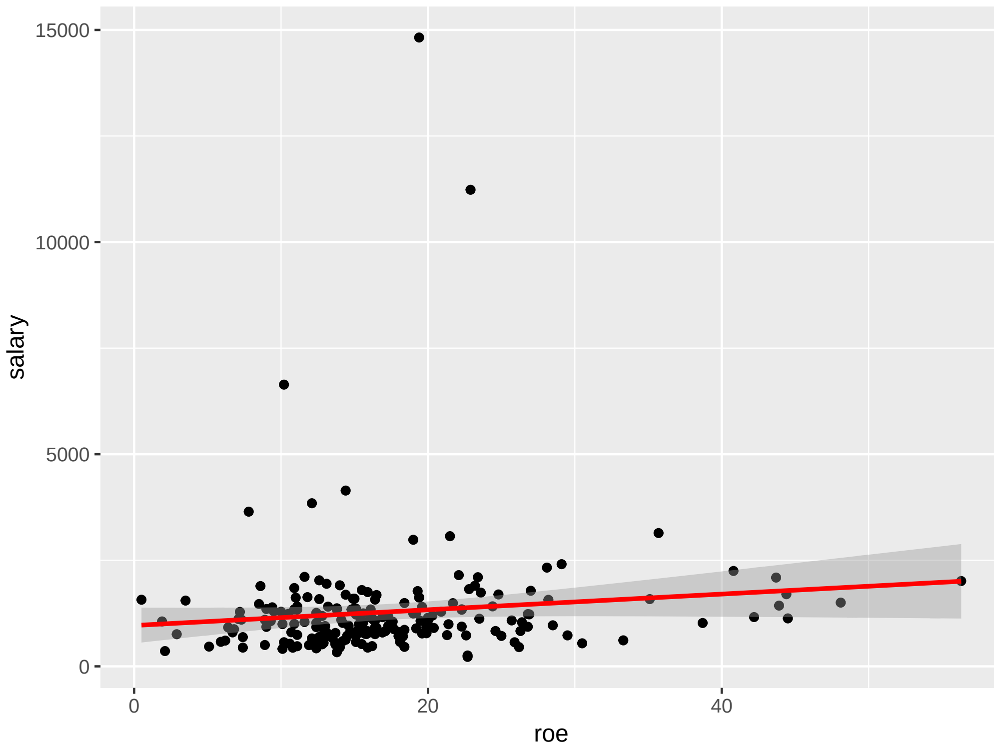


Figure 2.3: OLS regression ggplot2 style

```
## lm(formula = salary ~ roe, data = ceosal1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1160.2  -526.0  -254.0   138.8 13499.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   963.19     213.24   4.517 1.05e-05 ***
## roe            18.50      11.12   1.663   0.0978 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1367 on 207 degrees of freedom
## Multiple R-squared:  0.01319,    Adjusted R-squared:  0.008421
## F-statistic: 2.767 on 1 and 207 DF,  p-value: 0.09777
```

The `summary()` command also returns a list, and we can again use `names()` to learn what about the elements of this list.

```
names(summary(lm1_ceosal1))
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

So, for example, if we wanted to directly access the value of R^2, instead of copy and pasting it out of the printed statement from `summary()`, we could do so.

```r
summary(lm1_ceosal1)$r.squared
```

```
## [1] 0.01318862
```

Your turn

Recall that the residual sum of squares (SSR) is

$$R^2 = \frac{Var(\hat{y})}{Var(y)} = 1 - \frac{Var(\hat{u})}{Var(y)} \tag{2.3}$$

Calculate R^2 manually:

```r
var(fitted(lm1_ceosal1))/var(ceosal1$salary)
```

```
## [1] 0.01318862
```

```r
1 - var(residuals(lm1_ceosal1))/var(ceosal1$salary)
```

```
## [1] 0.01318862
```

Another useful function is the `predict()` function.

```r
set.seed(123)
roe_sample <-sample(ceosal1$roe, 1)
```

Let's make a prediction for salary when the return on equity is 20.2999992.

```r
b0hat_sample <- mean(salary) - b1hat * roe_sample
```

We are not restricted to observed values of the explanatory variable. Instead we can supply also our own predictor values

```r
predict(lm1_ceosal1, newdata = data.frame(roe = 30))
```

```
##        1
## 1518.227
```

The above code reads "predict the salary when the return on equity is 30 using the *lm1_ceosal1* model."

2.1.1 Regression through the Origin and Regression on a Constant

Regression without intercept (through origin)

```r
lm2 <- lm(salary ~  0 + roe, data = ceosal1)
```

Regression without slope

```r
lm3 <- lm(salary ~ 1, data = ceosal1)
```

```r
plot(salary~ roe, data = ceosal1,
     xlab = "Return on equity",
     ylab = "Salary",
     main = "Salary vs return on equity",
     pch  = 20,
     cex  = 2,
     col  = "grey")
```

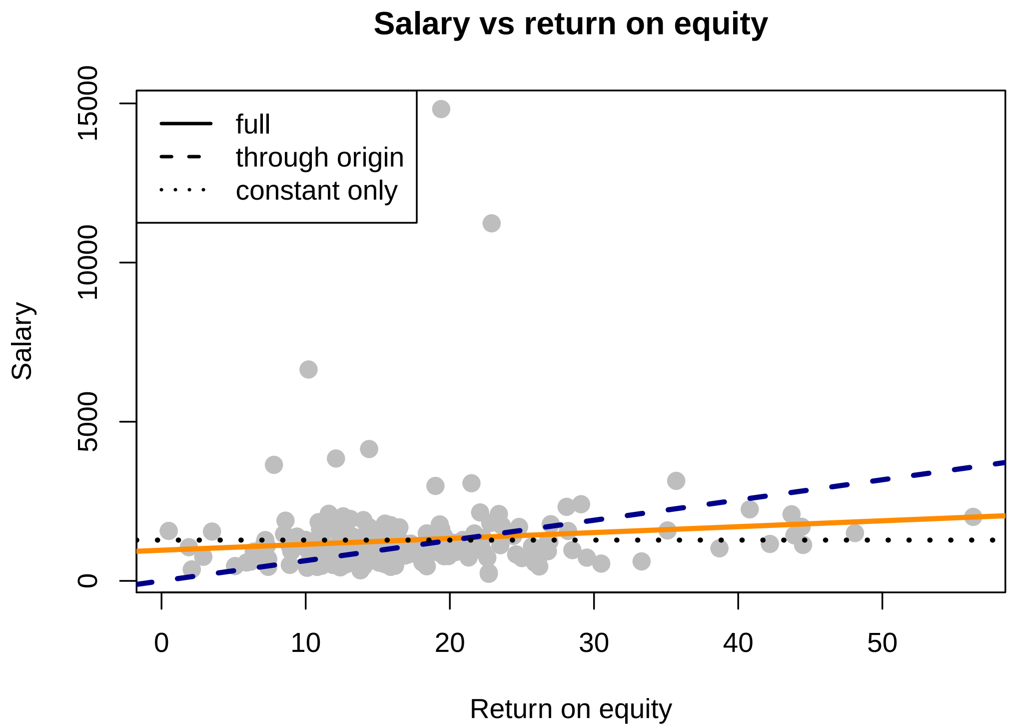


Figure 2.4: Regression through the Origin and on a Constant

```
abline(lm1_ceosal1, lwd = 3, lty = 1, col = "darkorange")
abline(lm2,lwd = 3,  lty = 2,   col = "darkblue")
abline(lm3, lwd = 3,  lty = 3,   col = "black")
legend("topleft",
       c("full",
         "through origin",
         "constant only"),
       lwd =2,
       lty = 1:3)
```

2.1.2 Simulating SLR

2.1.2.0.1 Expected Values, Variance, and Standard Errors

The **Gauss–Markov theorem** tells us that when estimating the parameters of the simple linear regression model β_0 and β_1, the $\hat{\beta}_0$ and $\hat{\beta}_1$ which we derived are the best linear unbiased estimates, or BLUE for short. (The actual conditions for the Gauss–Markov theorem are more relaxed than the SLR model.)

In short those assumptions are:

- SLR.1 Linear population regression function $y = \beta_0 + \beta_1 \times x + u$
- SLR.2 Random sampling of x and y from the population

- SLR.3 Variation in the sample values: $x_1, \ldots, x_n$
- SLR.4 Zero conditional mean: $\mathbf{E}(u|x) = 0$
- SLR.5 Homeskedasticity: $Var(u|x) = \sigma^2$

Recall that under **SLR.1 - SLR.4** the OLS parameter estimators are unbiased. Under **SLR.1 - SLR.4** the

OLS parameter estimators have a specific sampling variance.

Simulating a model is an important concept. In practice you will almost never have a true model, and you will use data to attempt to recover information about the unknown true model. With simulation, we decide the true model and simulate data from it. Then, we apply a method to the data, in this case least squares. Now, since we know the true model, we can assess how well it did.

Simulation also helps to grasp the concepts of estimators, estimates, unbiasedness, the sampling variance of the estimators, and the consequences of violated assumptions.

Sample size

```
n <- 200
```

True parameters

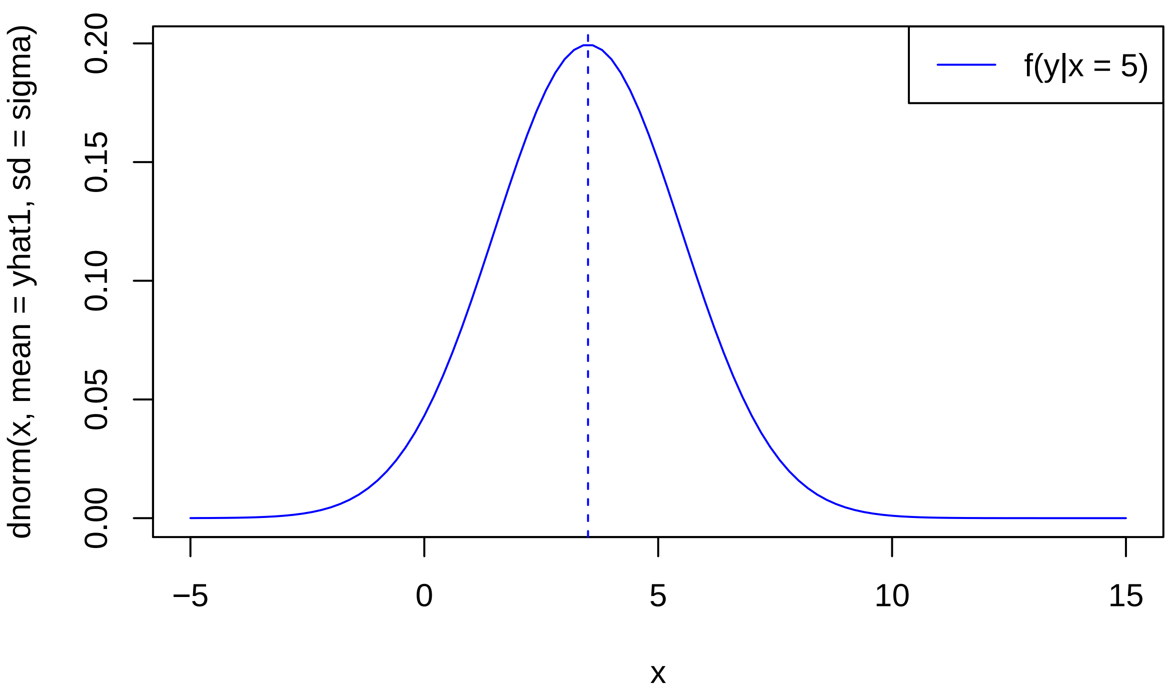
```
b0<- 1
b1 <- 0.5
sigma <- 2 # standard deviation of the error term u
x1 <- 5
```

Determine the distribution of the independent variable

```
yhat1 <- b0 + b1 * x1 #  Note that we do not include the error term
```

Plot a Gaussian distribution of the dependent variable based on the parameters

```
curve(dnorm(x, mean = yhat1, sd = sigma), -5, 15, col = "blue")
abline(v = yhat1, col = "blue", lty = 2)
legend("topright", legend = c("f(y|x = 5)"), lty = 1, col = c("blue"))
```



This represent the theoretical (true) probability distribution of y, given x

We can calculate the variance of b_1 and plot the corresponding density function.

$$var(b_2) = \frac{\sigma^2}{\sum(x_1 - \bar{x})} \tag{2.4}$$

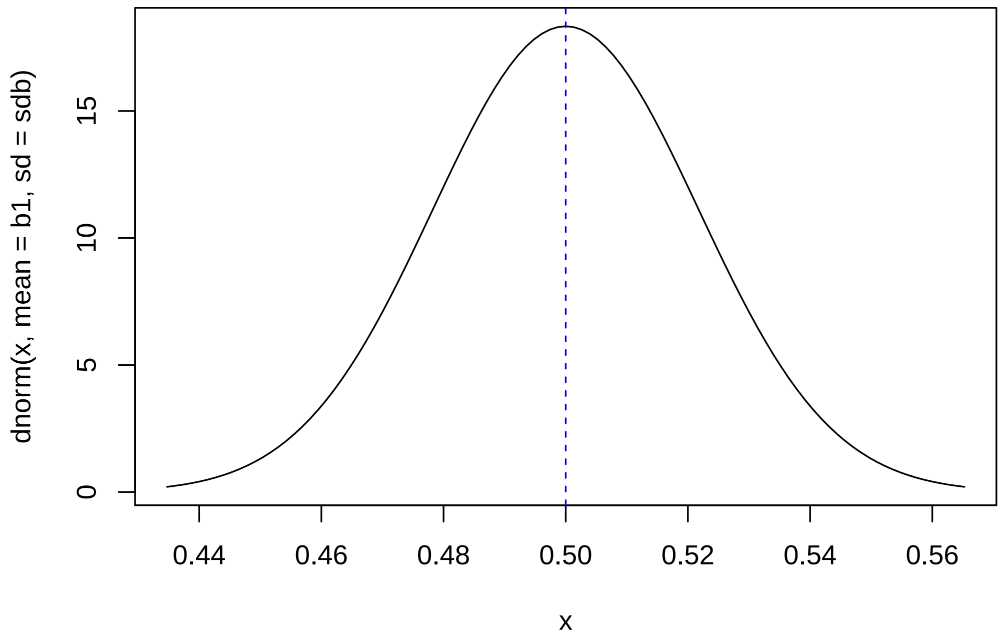


Figure 2.5: The theoretical (true) probability density function of b1

Assume that x_2 represents a second possible predictor of y

```
x2 <- 18

x <- c(rep(x1, n/2), rep(x2, n/2))
xbar <- mean(x)

sumxbar <- sum((x-xbar)^2)
varb <- (sigma^2)/sumxbar
sdb <-sqrt(varb)
leftlim <- b1-3*sdb
rightlim <- b1+3*sdb
```

```
curve(dnorm(x, mean = b1, sd = sdb), leftlim, rightlim,)
abline(v = b1, col = "blue", lty = 2)
```

Draw sample of size n

```
x <- rnorm(n, 4, sigma)
# Another way is to assume that the values for x are fixed and know
# x= seq(from = 0, to = 10, length.out = n)
```

```
u <- rnorm(n, 0, sigma)
```

```
y <- b0 + b1 * x + u
```

Estimate parameter by OLS

```
olsreg <- lm(y ~x )
```

```
simulation.df <- data.frame(x,y)
population.df <- data.frame(b0, b1)
```

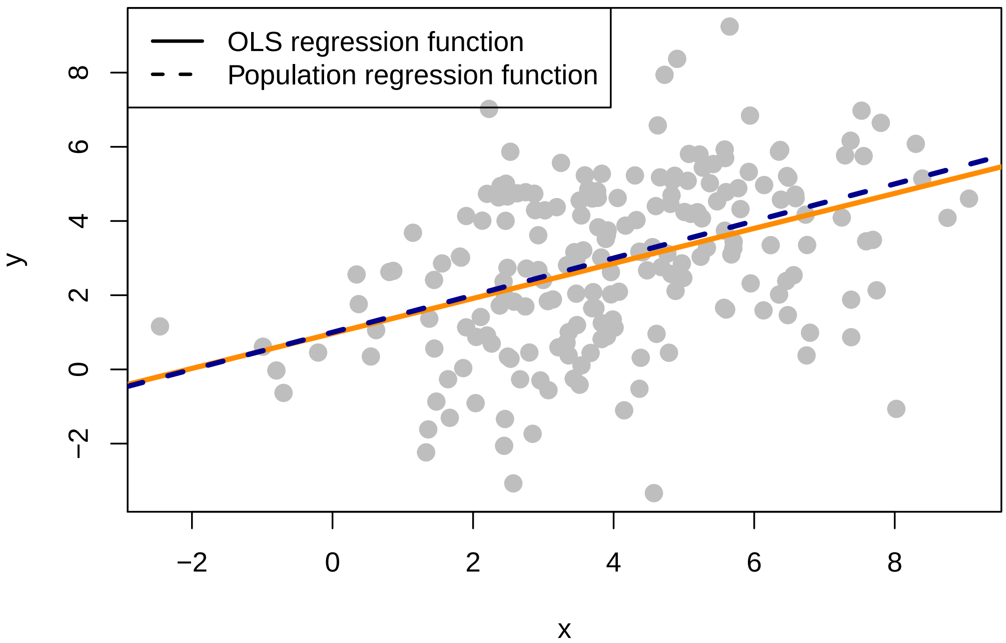


Figure 2.6: Simulated Sample and OLS Regression Line

```
plot(simulation.df,
     xlab = "x",
     ylab = "y",
     # main = "Simulate least squares regression",
     pch  = 20,
     cex  = 2,
     col  = "grey")
abline(olsreg, lwd = 3, lty = 1, col = "darkorange")
abline(b0, b1,  lwd = 3,  lty = 2,   col = "darkblue")
legend("topleft",
       c("OLS regression function",
         "Population regression function"),
       lwd =2,
       lty = 1:2)
```

```
lable1 <- "OLS regression function"
ggplot(simulation.df, aes(x = x,  y = y)) +
  geom_point() +
  geom_abline(aes(intercept=b0,slope=b1,colour="Population regression function"), linetype ="dashed", sl
  stat_smooth(aes(colour ="OLS regression function"), method = "lm",se=FALSE, show.legend =TRUE)+
  labs(colour = "Regression functions"
       # , title = "Simulate least squares regression"
  ) +
  theme_bw()
```

Since the expected values and variances of our estimators are defined over separate random samples from the same population, it makes sense to repeat our simulation exercise over many simulated samples.

```
# Set the random seed
set.seed(1234567)
```

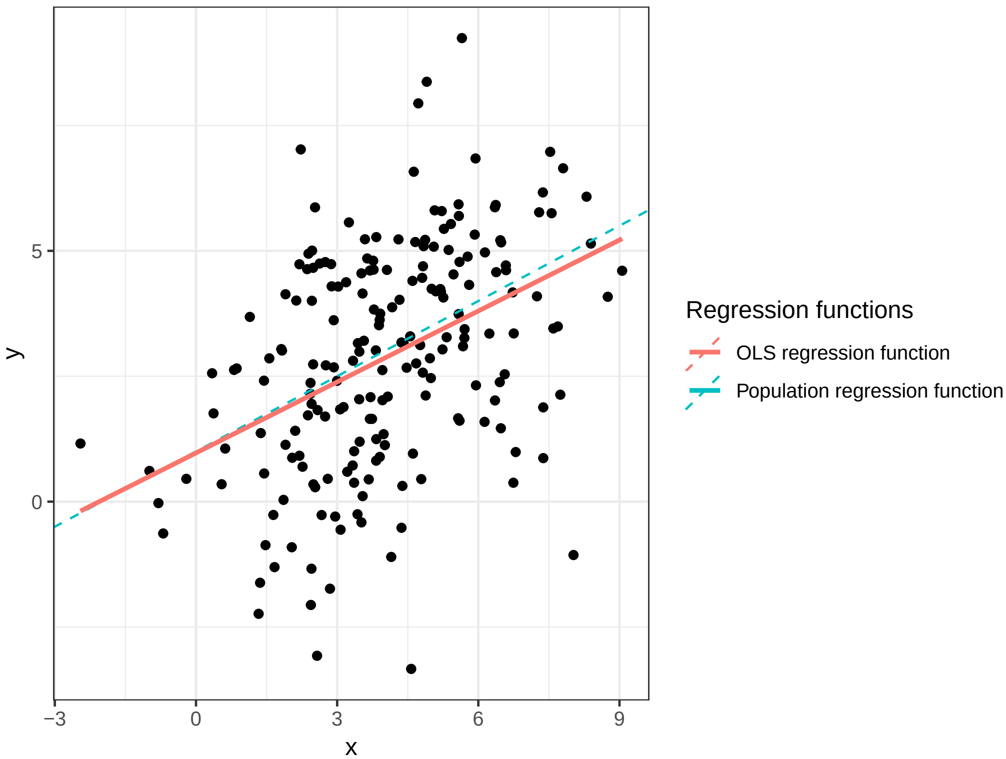


Figure 2.7: Simulated Sample and OLS Regression Line (gpplot Style)

```
# set sample size and number of simulations
n<-1000; r<-10000

# set true parameters: betas and sd of u
b0<-1.0; b1<-0.5; sigma<-2

# initialize b0hat and b1hat to store results later:
b0hat <- numeric(r)
b1hat <- numeric(r)

# Draw a sample of x, fixed over replications:
x <- rnorm(n,4,1)

# repeat r times:
for(j in 1:r) {
  # Draw a sample of y:
  u <- rnorm(n,0,sigma)
  y <- b0 + b1*x + u

  # estimate parameters by OLS and store them in the vectors
  bhat <- coefficients( lm(y~x) )
  b0hat[j] <- bhat["(Intercept)"]
  b1hat[j] <- bhat["x"]
}
```

```
# MC estimate of the expected values:
mean(b0hat)
```

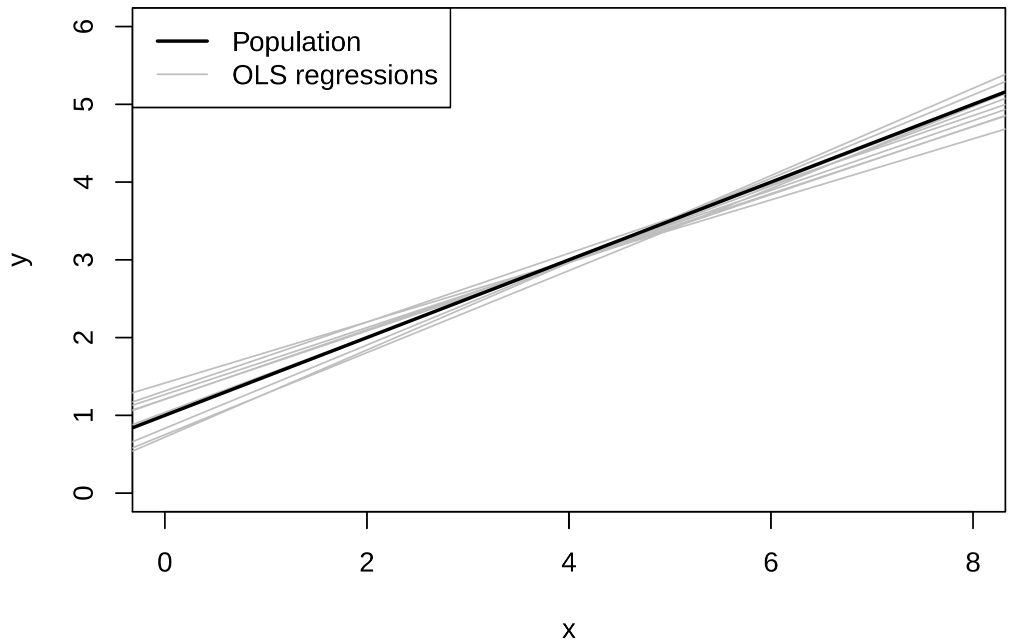


Figure 2.8: Population and Simulated OLS Regression Lines

```
## [1] 0.9985388
```

```
mean(b1hat)
```

```
## [1] 0.5000466
```

```
# MC estimate of the variances:
var(b0hat)
```

```
## [1] 0.0690833
```

```
var(b1hat)
```

```
## [1] 0.004069063
```

```
# Initialize empty plot
plot( NULL, xlim=c(0,8), ylim=c(0,6), xlab="x", ylab="y")
# add OLS regression lines
for (j in 1:10) abline(b0hat[j],b1hat[j],col="gray")
# add population regression line
abline(b0,b1,lwd=2)
# add legend
legend("topleft",c("Population","OLS regressions"),
       lwd=c(2,1),col=c("black","gray"))
```

Even though the loop solution is transparent, let us take a look at a different, more *modern* approach.

```
# define a function the returns the alpha -- its point estimate, standard error, etc. -- from the OLS
x <- rnorm(n,4,1) # NOTE 1: Although a normal distribution is usually defined by its mean and variance,
# NOTE 2: We use the same values for x in all samples since we draw them outside of the loop.

iteration <- function() {
  u <- rnorm(n,0,sigma)
  y <- b0 + b1*x + u
```

```
  lm(y~x) %>%
    broom::tidy() # %>%
  # filter(term == 'x') # One could only extract the slope
}

# 1000 iterations of the above simulation
MC_coef<- map_df(1:1000, ~iteration())
str(MC_coef)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2000 obs. of  5 variables:
##  $ term     : chr  "(Intercept)" "x" "(Intercept)" "x" ...
##  $ estimate : num  1.577 0.372 1.44 0.387 1.355 ...
##  $ std.error: num  0.2672 0.0639 0.2623 0.0628 0.2626 ...
##  $ statistic: num  5.9 5.82 5.49 6.17 5.16 ...
##  $ p.value  : num  4.94e-09 7.91e-09 5.13e-08 9.92e-10 2.99e-07 ...
```

Instead of plotting simulated and true parameter regression lines we can take a look at the kernel density of the simulated parameter estimates

Figure **??** shows the simulated distribution of β_0 and β_1 the theoretical one.

```
# plot the results
str(MC_coef)
```


```
## Classes 'tbl_df', 'tbl' and 'data.frame':    2000 obs. of  5 variables:
##  $ term     : chr  "(Intercept)" "x" "(Intercept)" "x" ...
##  $ estimate : num  1.577 0.372 1.44 0.387 1.355 ...
##  $ std.error: num  0.2672 0.0639 0.2623 0.0628 0.2626 ...
##  $ statistic: num  5.9 5.82 5.49 6.17 5.16 ...
##  $ p.value  : num  4.94e-09 7.91e-09 5.13e-08 9.92e-10 2.99e-07 ...
```


```
MC_coef<- MC_coef %>%
  mutate(OLScoeff =  ifelse(term == "x", "b1hat", "b0hat")) %>%  # rename the x to b1hat and (Intercept
  mutate(Simulated = ifelse(term == "x", "b1", "b0")) #  %>%
```

```
ggplot(data= MC_coef, aes(estimate)) +
  geom_histogram() +
  geom_vline(data = filter(MC_coef, OLScoeff == "b0hat"), aes(xintercept=b0), colour="pink") +
  geom_vline(data = filter(MC_coef, OLScoeff == "b1hat"), aes(xintercept=b1), colour="darkgreen") +
  geom_text(data=MC_coef[3,], mapping=aes(x=estimate, y=8, label=paste("True parameter: ", MC_coef[3,7]
  geom_text(data=MC_coef[4,], mapping=aes(x=estimate, y=8, label=paste("True parameter: ", MC_coef[4,7]
  facet_wrap( ~ OLScoeff, scales = "free")   +
  labs(
    title = "Histogram Monte Carlo Simulations and True population parameters") +
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
b1_sim <- MC_coef %>%
  filter(Simulated == "b1")

mean(b1_sim$estimate)
```

```
## [1] 0.5011414
```

```
var(b1_sim$estimate) == (sd(b1_sim$estimate))^2
```

```
## [1] FALSE
```

Figure 2.9: Histogram b0 and b1 and true parameter

```
all.equal(var(b1_sim$estimate) , (sd(b1_sim$estimate))^2) # Floating point arithmetic!
```

```
## [1] TRUE
```

```
ggplot(data= b1_sim, aes(estimate)) +
  geom_density(aes(fill = Simulated), alpha = 0.2) + # computes and draws the kernel density, which is
  # stat_function(fun = dnorm, args = list(mean = mean(b1_sim$estimate), sd = sd(b1_sim$estimate)), aes
  stat_function(fun = dnorm, args = list(mean = 0.5, sd = sd(b1_sim$estimate)), aes(colour = "true")) +
  # labs(
  #  title = "Kernel Density Monte Carlo Simulations vs. True population parameters"
  # ) +
  scale_color_discrete(name="") +
  theme_bw()
```

Rework this section might have mixed up what is simulated and what is biased

2.1.2.0.2 Violation of SLR.4

To implement a violation of **SLR.4** (zero conditional mean) consider a case where in the population u is not mean independent of x, for example

$$\mathbf{E}(u|x) = \frac{x-4}{5}$$

```
# Set the random seed
set.seed(1234567)

# set sample size and number of simulations
```

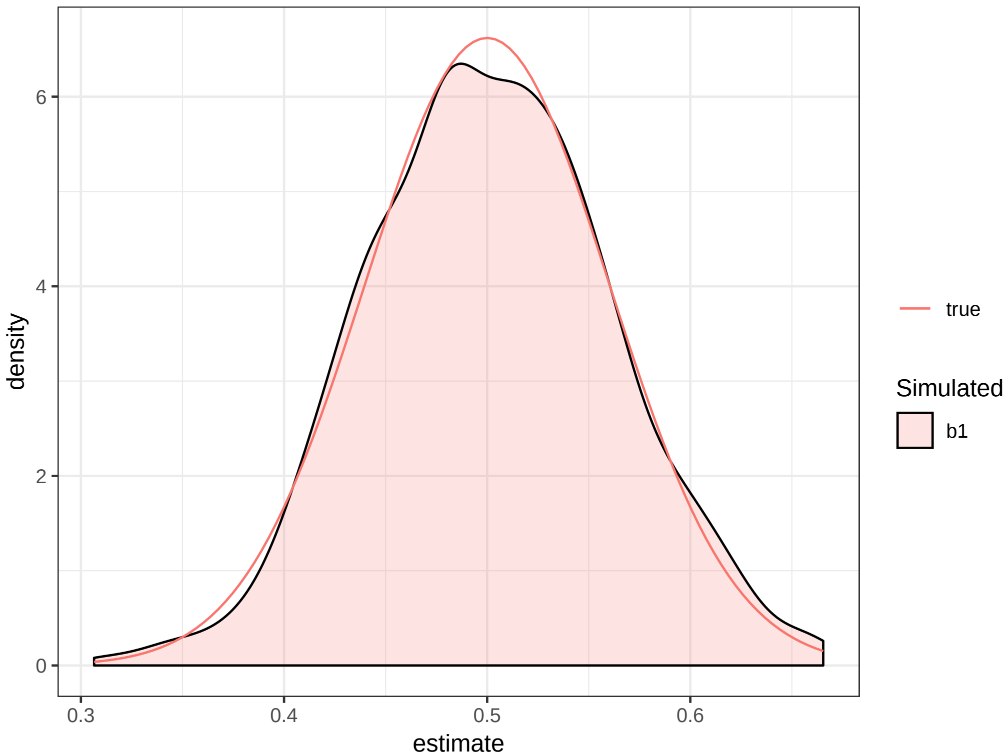


Figure 2.10: Simulated and theoretical distributions of b1

```
n<-1000; r<-10000

# set true parameters: betas and sd of u
b0<-1; b1<-0.5; su<-2

# initialize b0hat and b1hat to store results later:
b0hat <- numeric(r)
b1hat <- numeric(r)

# Draw a sample of x, fixed over replications:
x <- rnorm(n,4,1)

# repeat r times:
for(j in 1:r) {
  # Draw a sample of y:
  u <- rnorm(n, (x-4)/5, su) # this is where manipulate the assumption of zero conditional mean
  y <- b0 + b1*x + u

  # estimate parameters by OLS and store them in the vectors
  bhat <- coefficients( lm(y~x) )
  b0hat[j] <- bhat["(Intercept)"]
  b1hat[j] <- bhat["x"]
}
```

OLS coefficients

```
# MC estimate of the expected values:
mean(b0hat)
```

```
## [1] 0.1985388
```

```
mean(b1hat)
```

```
## [1] 0.7000466
```

```
# MC estimate of the variances:
var(b0hat)
```

```
## [1] 0.0690833
```

```
var(b1hat)
```

```
## [1] 0.004069063
```

The average estimates are far from the population parameters $\beta_0 = 1$ and $\beta_1 = 0.5$!

2.1.2.0.3 Violation of SLR.5

Homoskedasticity is not required for unbiasedness but for it is a requirement for the theorem of sampling variance. Consider the following heteroskedastic behavior of u given x.

```
# Set the random seed
set.seed(1234567)

# set sample size and number of simulations
n<-1000; r<-10000

# set true parameters: betas and sd of u
b0<-1; b1<-0.5; su<-2

# initialize b0hat and b1hat to store results later:
b0hat <- numeric(r)
b1hat <- numeric(r)

# Draw a sample of x, fixed over replications:
x <- rnorm(n,4,1)

# repeat r times:
for(j in 1:r) {
  # Draw a sample of y:
  varu <- 4/exp(4.5) * exp(x)
  u <- rnorm(n, 0, sqrt(varu) )
  y <- b0 + b1*x + u

  # estimate parameters by OLS and store them in the vectors
  lm_heterosced <- lm(y~x)

  bhat <- coefficients( lm(y~x) )
  b0hat[j] <- bhat["(Intercept)"]
  b1hat[j] <- bhat["x"]
}
```

```
summary(lm_heterosced) # just the last sample of the MC-simulation
```

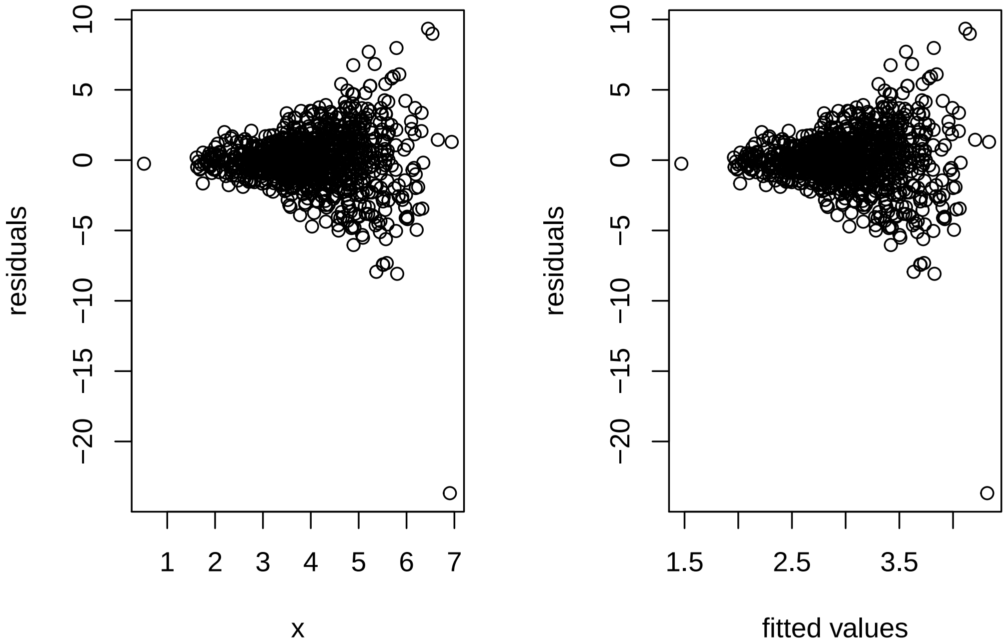


Figure 2.11: Heteroskedasticity in the simulated data

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.6742  -0.9033   0.0052   1.0012   9.3411
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.24088    0.27158   4.569 5.51e-06 ***
## x            0.44561    0.06593   6.759 2.37e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.075 on 998 degrees of freedom
## Multiple R-squared:  0.04377,    Adjusted R-squared:  0.04281
## F-statistic: 45.68 on 1 and 998 DF,  p-value: 2.367e-11
```

Plot the residual against the regressor suspected of creating heteroskedasticity, or more generally, the fitted values of the regression.

```
res <- residuals(lm_heterosced)
yhat <- fitted(lm_heterosced)
```

```
par(mfrow = c(1,2))
plot(x, res, ylab = "residuals")
plot(yhat, res, xlab = "fitted values", ylab = "residuals")
```

```
# MC estimate of the expected values:
mean(b0hat)
```

```
## [1] 1.0019
```

```
mean(b1hat)
```

```
## [1] 0.4992376
```

```
# MC estimate of the variances:
var(b0hat)
```

```
## [1] 0.08967037
```

```
var(b1hat)
```

```
## [1] 0.007264373
```

Unbiasedness is provided but sampling variance is incorrect (compared to the results provided above).

2.1.3 Nonlinearities

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

We will use a new dataset, *wage1*, for this section. A detailed exploratory analysis of the dataset is left to the reader.

```
data("wage1")
attach(wage1)
```

```
## The following objects are masked from wage1 (pos = 14):
##
##     clerocc, construc, educ, exper, expersq, female, lwage,
##     married, ndurman, nonwhite, northcen, numdep, profocc,
##     profserv, services, servocc, smsa, south, tenure, tenursq,
##     trade, trcommpu, wage, west
```

2.1.3.1 Predicated variable transformation

A common variance stabilizing transformation (VST) when we see increasing variance in a fitted versus residuals plot is $log(Y)$.

Related, to use the *log* of an independent variable is to make its distribution closer to the normal distribution.

```
# wage1$logwage <- log(wage1$wage) # one could also create a new variable

p1_wagehisto <- ggplot(wage1)  +
  geom_histogram(aes(x = wage), fill = "red", alpha = 0.6) +
  theme_bw()


p2_wagehisto <- ggplot(wage1)  +
  geom_histogram(aes(x = wage),  fill = "blue", alpha = 0.6) +
  scale_x_continuous(trans='log2', "Log Wage") + # instead of creating a new variable with simply defin
  theme_bw()
```

```
ggarrange(p1_wagehisto, p2_wagehisto,
          labels = c("A", "B"),
          ncol = 2, nrow = 1)
```

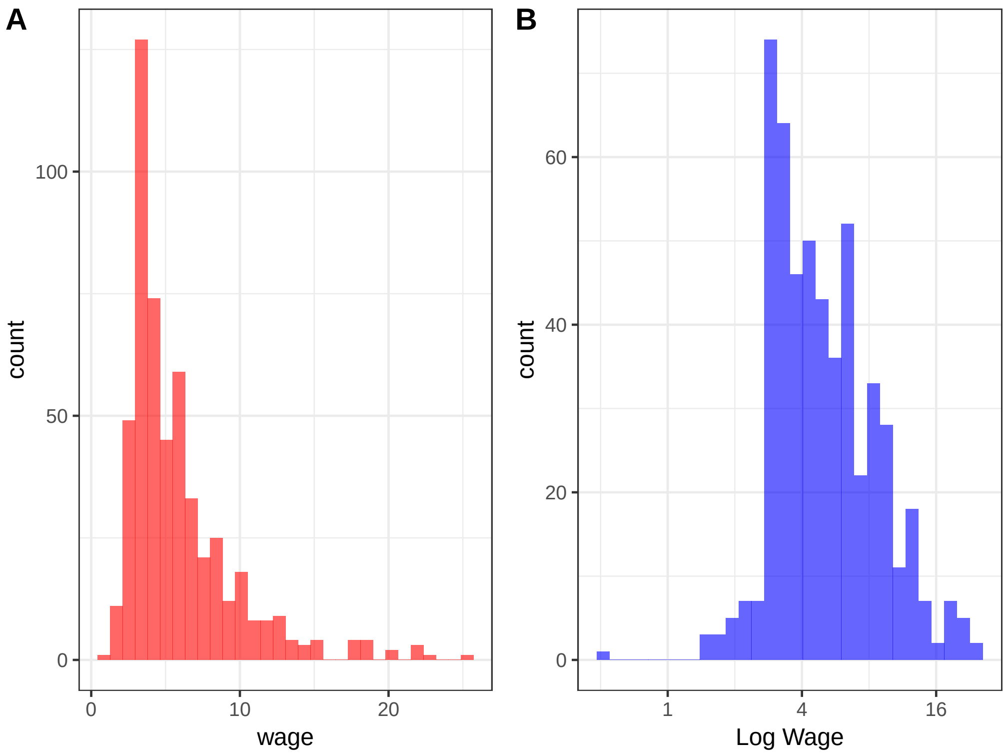


Figure 2.12: Histogram of wage and log(wage)

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

A model with a log transformed response:

$$log(Y_i) = \beta_0 + \beta_1 \times x_i + \epsilon_i \tag{2.5}$$

```
lm_wage <- lm(wage ~ educ, data = wage1)
lm_wage1 <- lm(log(wage)~ educ, data =  wage1)
summary(lm_wage)
```

```
##
## Call:
## lm(formula = wage ~ educ, data = wage1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.3396 -2.1501 -0.9674  1.1921 16.6085
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.90485    0.68497  -1.321    0.187
## educ         0.54136    0.05325  10.167   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.378 on 524 degrees of freedom
```

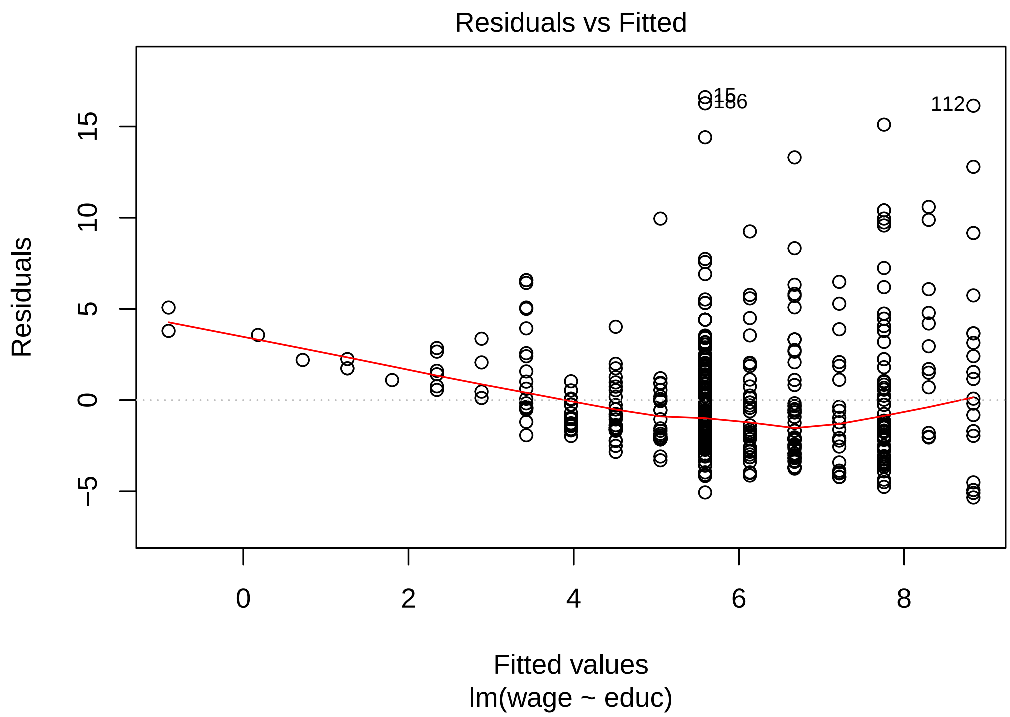


Figure 2.13: Regression diagnostics plot base R - Linear Relationship

```
## Multiple R-squared:  0.1648, Adjusted R-squared:  0.1632
## F-statistic: 103.4 on 1 and 524 DF,  p-value: < 2.2e-16
```

```
summary(lm_wage1)
```

```
##
## Call:
## lm(formula = log(wage) ~ educ, data = wage1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.21158 -0.36393 -0.07263  0.29712  1.52339
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.583773   0.097336   5.998 3.74e-09 ***
## educ        0.082744   0.007567  10.935  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4801 on 524 degrees of freedom
## Multiple R-squared:  0.1858, Adjusted R-squared:  0.1843
## F-statistic: 119.6 on 1 and 524 DF,  p-value: < 2.2e-16
```

Plotting Diagnostics for Linear Models

```
plot(lm_wage)
```

```
autoplot(lm_wage, which = 1:6, colour = 'dodgerblue3',
         smooth.colour = 'red', smooth.linetype = 'dashed',
```

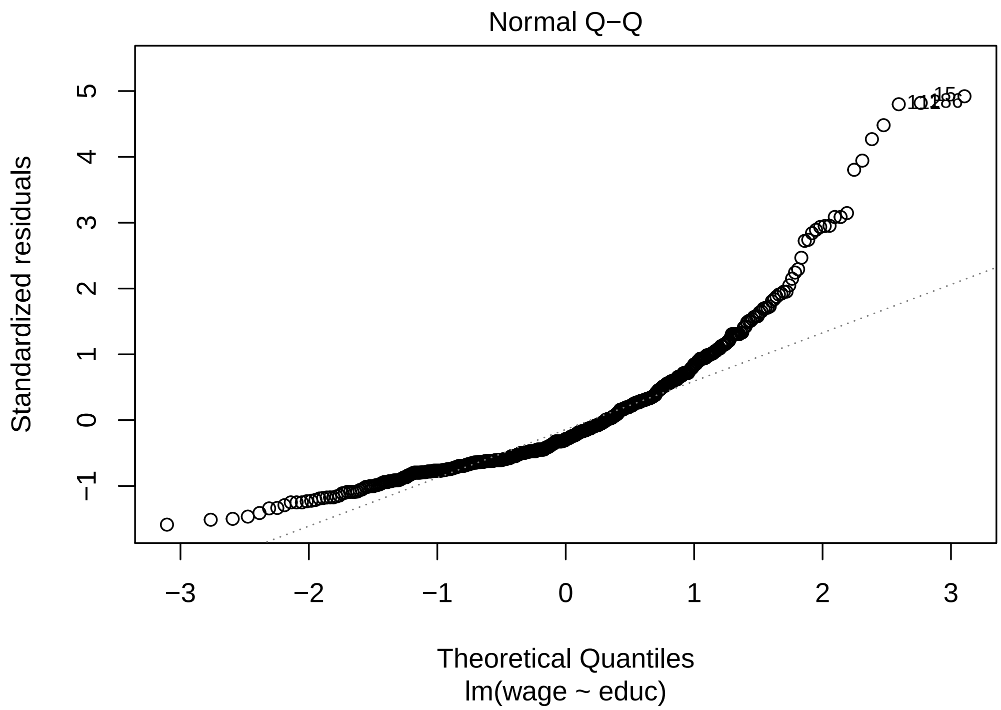


Figure 2.14: Regression diagnostics plot base R - Linear Relationship

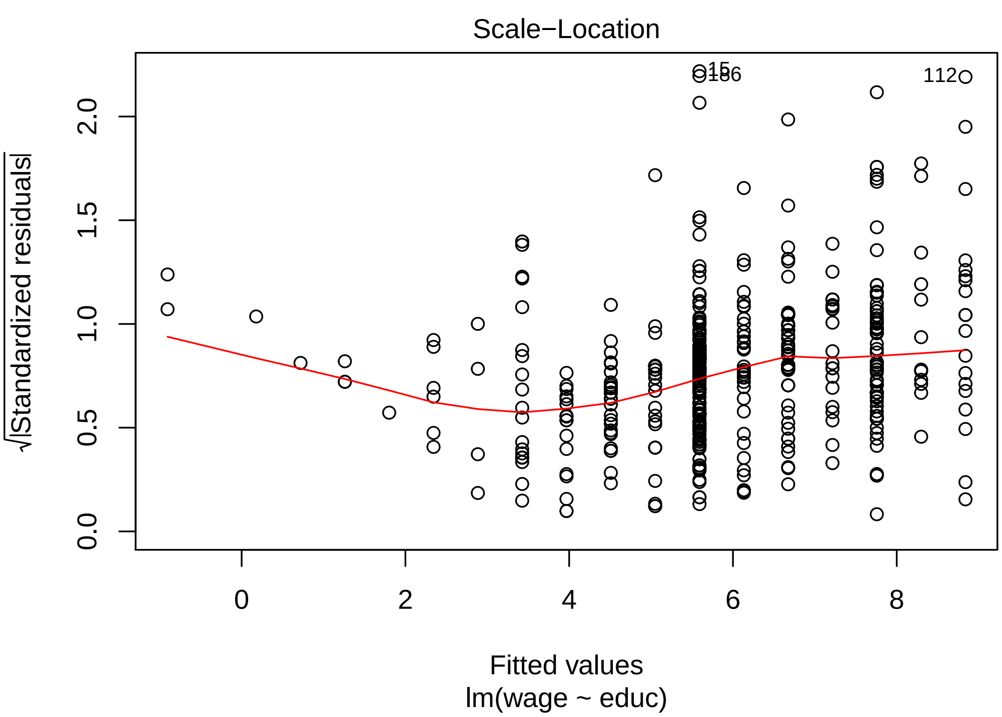


Figure 2.15: Regression diagnostics plot base R - Linear Relationship

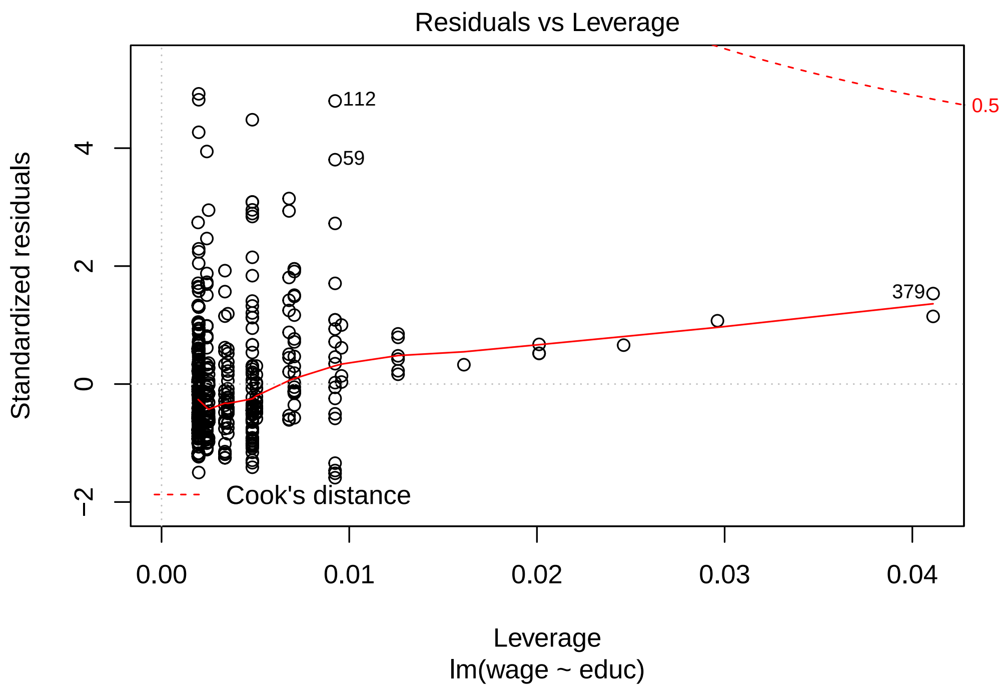


Figure 2.16: Regression diagnostics plot base R - Linear Relationship

```
         ad.colour = 'blue',
         label = FALSE,
         label.size = 3, label.n = 5, label.colour = 'blue',
         ncol = 3) +
  theme_bw()
```

```
autoplot(lm_wage1, which = 1:6, colour = 'dodgerblue3',
         smooth.colour = 'red', smooth.linetype = 'dashed',
         ad.colour = 'blue',
         label = FALSE,
         label.size = 3, label.n = 5, label.colour = 'blue',
         ncol = 3) +
  theme_bw()
```

```
p1_nonlinearities <- ggplot(wage1, aes(x = educ, y = wage )) +
  geom_point()   +
  scale_y_continuous(trans='log2', "Log Wage") +
  stat_smooth(aes(fill="Linear Model"),size=1,method = "lm" ,span =0.3, se=F) +
  guides(fill = guide_legend("Model Type")) +
  theme_bw()
```

Note that if we re-scale the model from a log scale back to the original scale of the data, we now have

$$Y_i = exp(\beta_0 + \beta_1 \times x_i) \times exp(\epsilon_i) \tag{2.6}$$

which has errors entering in a multiplicative fashion.

```
log.model.df <- data.frame(x = wage1$educ,
                           y = exp(fitted(lm_wage1))) # This is essentially exp(b0_wage1 + b1_wage1 * w
```

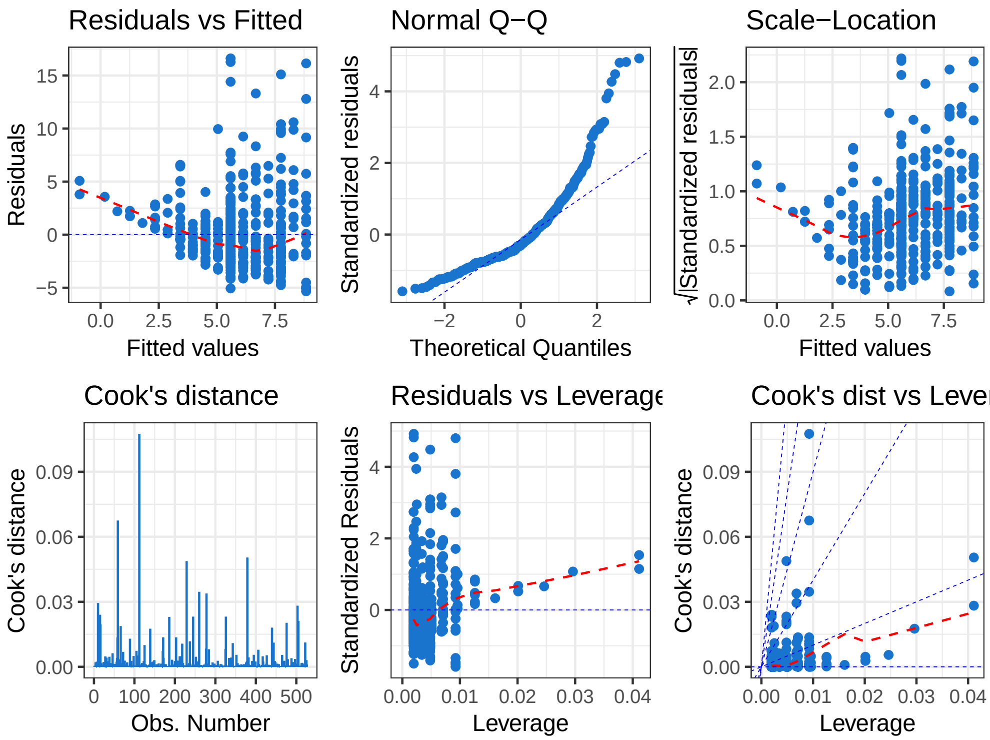


Figure 2.17: Regression diagnostics autoplot(ggplot) - Linear Relationship

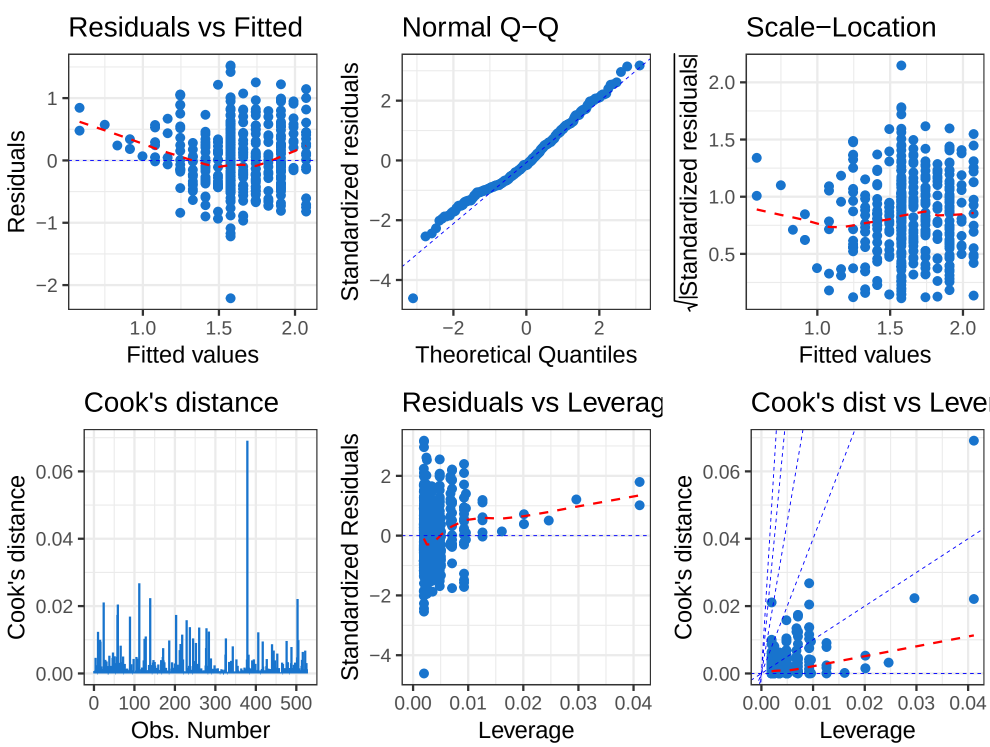


Figure 2.18: Regression diagnostics - Non-Linear Relationship

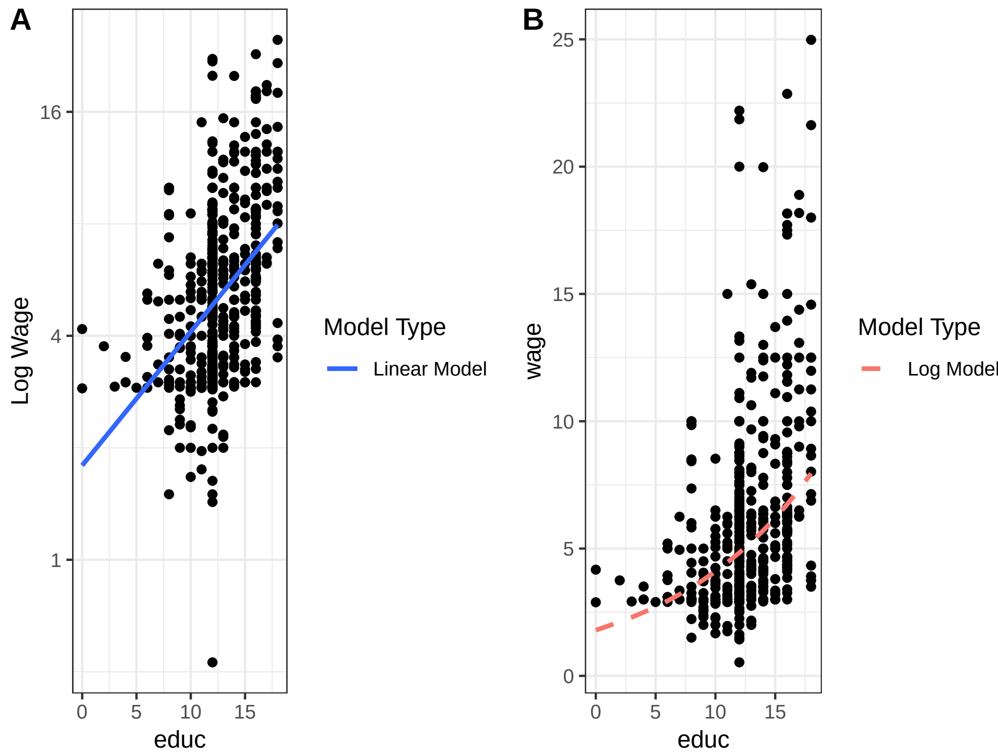


Figure 2.19: Wages by Education - Different transformations

```
p2_nonlinearities <- ggplot(wage1, aes(x = educ, y = wage))  +
  geom_point()   +
  geom_line(data = log.model.df, aes(x, y, color = "Log Model"), size = 1, linetype = 2)  +
  guides(color = guide_legend("Model Type")) +
  theme_bw()
```

```
ggarrange(p1_nonlinearities, p2_nonlinearities,
          labels = c("A", "B"),
          ncol = 2, nrow = 1)
```

A: Plotting the data on the transformed log scale and adding the fitted line, the relationship again appears linear, and the variation about the fitted line looks more constant.

B: By plotting the data on the original scale, and adding the fitted regression, we see an exponential relationship. However, this is still a *linear* model, since the new transformed response, $log(Y_i$, is still a *linear* combination of the predictors. In other words, only β needs to be linear, not the x values.

NOTE:

The example comes from the Wooldrige book but the variable educ looks more like count data. A Poisson GLM might seems like a better choice.

Quadratic Model

$$Y_i = \beta_0 + \beta_1 \times x_i^2) \times \epsilon_i \tag{2.7}$$

New dataset from Wooldrige: Collected from the real estate pages of the Boston Globe during 1990. These are homes that sold in the Boston, MA area.

```
data("hprice1")
attach(hprice1)
```

```
## The following objects are masked from hprice1 (pos = 14):
##
##     assess, bdrms, colonial, lassess, llotsize, lotsize, lprice,
##     lsqrft, price, sqrft
```

In R, independent variables involving mathematical operators can be included in regression equation with the function `I()`

```
lm_hprice <- lm(price ~ sqrft, data  = hprice1)
lm_hprice1 <- lm(price ~ sqrft + I(sqrft^2), data  = hprice1)
```

Alternatively use the `poly()` function. Be careful of the additional argument `raw`.

```
lm_hprice2 <- lm(price ~ poly(sqrft, degree = 2),  data  = hprice1)
lm_hprice3 <- lm(price ~ poly(sqrft, degree = 2, raw = TRUE),  data  = hprice1) # if true, use raw and
```

```
unname(coef(lm_hprice1))
```

```
## [1]  1.849453e+02 -1.710855e-02  3.262809e-05
```

```
unname(coef(lm_hprice2))
```

```
## [1] 293.5460 754.8517 135.6051
```

```
unname(coef(lm_hprice3))
```

```
## [1]  1.849453e+02 -1.710855e-02  3.262809e-05
```

```
all.equal(unname(coef(lm_hprice1)), unname(coef(lm_hprice2)))
```

```
## [1] "Mean relative difference: 5.401501"
```

```
all.equal(unname(coef(lm_hprice1)), unname(coef(lm_hprice3)))
```

```
## [1] TRUE
```

```
all.equal(fitted(lm_hprice1), fitted(lm_hprice2))
```

```
## [1] TRUE
```

```
all.equal(fitted(lm_hprice1), fitted(lm_hprice3))
```

```
## [1] TRUE
```

2.1.4 Inference for Simple Linear Regression

"There are three types of lies: lies, damn lies, and statistics" *Benjamin Disraeli*

2.2 Multiple Linear Regression

Note

A **(general) linear model** is similar to the simple variant, but with a multivariate $x \epsilon R^{\rho}$ and a mean given by a hyperplane in place of a single line.

- General principles are the same as the simple case
- Math is more difficult because we need to use matrices
- Interpretation is more difficult because the β_j are effects conditional on the other variables

Many would retain the same signs as the simple linear regression, but the magnitudes would be smaller. In some cases, it is possible for the relationship to flip directions when a second (highly correlated) variable is added.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + u \tag{2.8}$$

The next example from Wooldrige relates the college GPA ($cloGPA$) to the high school GPA ("hsGPA") and achievment test score (ACT) for a sample of 141 students.

```
data("gpa1")
attach(gpa1)
```

```
## The following object is masked from package:robustbase:
##
##     alcohol

## The following object is masked from bwght:
##
##     male

## The following objects are masked from gpa1 (pos = 14):
##
##     ACT, age, alcohol, bgfriend, bike, business, campus, car,
##     clubs, colGPA, drive, engineer, fathcoll, gradMI, greek,
##     hsGPA, job19, job20, junior, male, mothcoll, PC, senior,
##     senior5, siblings, skipped, soph, voluntr, walk

## The following objects are masked from package:wooldridge:
##
##     alcohol, campus
```

Obtain parameter estimates

```
GPAres <- lm(colGPA ~ hsGPA + ACT, data = gpa1)
summary(GPAres)
```

```
##
## Call:
## lm(formula = colGPA ~ hsGPA + ACT, data = gpa1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.85442 -0.24666 -0.02614  0.28127  0.85357
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.286328   0.340822   3.774 0.000238 ***
## hsGPA       0.453456   0.095813   4.733 5.42e-06 ***
## ACT         0.009426   0.010777   0.875 0.383297
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.3403 on 138 degrees of freedom
## Multiple R-squared:  0.1764, Adjusted R-squared:  0.1645
## F-statistic: 14.78 on 2 and 138 DF,  p-value: 1.526e-06
```

```r
coef(GPAres)[[1]]
```

```
## [1] 1.286328
```

In the multiple linear regression setting, some of the interpretations of the coefficients change slightly. Here, $\hat{\beta}_0 = 1.2863278$ is our estimate for β_0 when all of the predictors are 0. In this example this makes sense but think of the following example:

Your turn

Assume the following model:

```r
mpg_model = lm(hp ~ wt + cyl, data = mtcars)
coef(mpg_model)
```

```
## (Intercept)          wt         cyl
##  -51.805567    1.330463   31.387901
```

How do you interpret the intercept coefficient estimate?

A: Here, $\hat{\beta}_0$ = -51.8055669 is our estimate for β_0, the mean gross horsepower for a car that weights 0 pounds and has 0 cylinders. We see our estimate here is negative, which is a physical impossibility. However, this isn't unexpected, as we shouldn't expect our model to be accurate for cars which weight 0 pounds and have no cylinders to propel the engine.

```r
with (gpa1, {
  # find min-max seq for grid construction
  min_hsGPA <- min(gpa1$hsGPA)
  max_hsGPA <- max(gpa1$hsGPA)
  min_ACT <- min(gpa1$ACT)
  max_ACT <- max(gpa1$ACT)


  # linear regression
  fit <- lm(colGPA ~ hsGPA + ACT)

  # predict values on regular xy grid
  hsGPA.pred <- seq(min_hsGPA, max_hsGPA, length.out = 30)
  ACT.pred <- seq(min_ACT, max_ACT, length.out = 30)
  xy <- expand.grid(hsGPA = hsGPA.pred,
                    ACT = ACT.pred)

  colGPA.pred <- matrix (nrow = 30, ncol = 30,
                         data = predict(fit, newdata = data.frame(xy),
                                        interval = "prediction"))

  # fitted points for droplines to surface
  fitpoints <- predict(fit)

  scatter3D(z = colGPA, x = hsGPA, y = ACT, pch = 18, cex = 2,
            theta = 20, phi = 20, ticktype = "detailed",
```

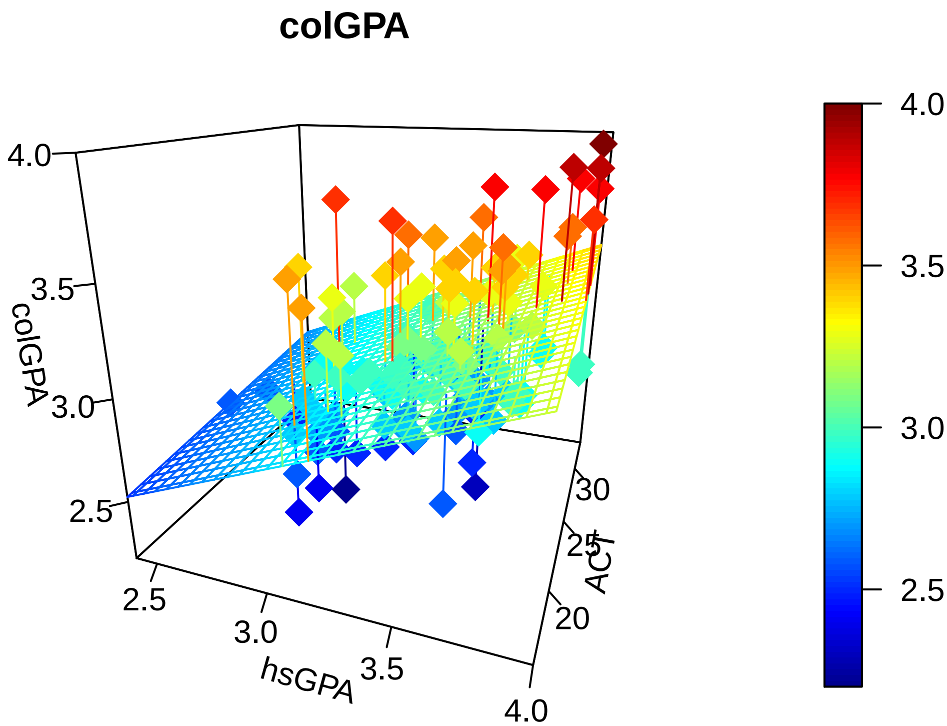


Figure 2.20: College GPA High School GPA + Achievment test score

```
            xlab = "hsGPA", ylab = "ACT", zlab = "colGPA",
            surf = list(x = hsGPA.pred, y = ACT.pred, z = colGPA.pred,
                        facets = NA, fit = fitpoints),
            main = "colGPA")

})
```

The data points (x_{i1},x_{i2},y_i) now exist in 3-dimensional space, so instead of fitting a line to the data, we will fit a plane.

2.2.1 Ceteris Paribus Interpretation and Omitted Variable bias

Consider a regression with two explanatory variables

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \tag{2.9}$$

```
# Parameter estimates for full and simple model:
beta.hat <- coef( lm(colGPA ~ ACT+hsGPA, data=gpa1) )
beta.hat
```

```
## (Intercept)         ACT       hsGPA
## 1.286327767 0.009426012 0.453455885
```

Now, lets ommit one variable in the regression

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 \tag{2.10}$$

```r
# Relation between regressors:
delta.tilde <- coef( lm(hsGPA ~ ACT, data=gpa1) )
delta.tilde
```

```
## (Intercept)         ACT
##  2.46253658  0.03889675
```

The parameter $\hat{\beta}_1$ is the estimated effect of increasing x_1 by one unit (and **NOT** keeping x_2 fixed). It can be related to $\hat{\beta}_1$ using the formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 \tilde{\delta}_1 \tag{2.11}$$

where

$$\hat{x}_2 = \tilde{\delta}_0 + \tilde{\delta}_1 x_1 \tag{2.12}$$

```r
# Omitted variables formula for beta1.tilde:
beta.hat["ACT"] + beta.hat["hsGPA"]*delta.tilde["ACT"]
```

```
##        ACT
## 0.02706397
```

```r
# Actual regression with hsGPA omitted:
lm(colGPA ~ ACT, data=gpa1)
```

```
##
## Call:
## lm(formula = colGPA ~ ACT, data = gpa1)
##
## Coefficients:
## (Intercept)          ACT
##     2.40298      0.02706
```

In this example, the indirect effect is actually stronger than the direct effect. *ACT* predicts *colGPA* mainly because it is related to *hsGPA* which in turn is strongly related to *colGPA*.

2.2.2 Standard errors, Multicollinearity and VIF

We know already how we can extract the standard errors

```r
GPAres <- lm(colGPA ~ hsGPA + ACT, data = gpa1)
SER<-summary(GPAres)$sigma
```

The variance inflation factor, **VIF**, accounts for (imperfect) multicollinearity.If x_t is highly related to the other regressors, R_j^2 and therefore also VIF_j and the variance of $\hat{\beta}_j$ are large.

$$\frac{1}{1 - R_j^2} \tag{2.13}$$

```r
# regressing hsGPA on ACT for calculation of R2 & VIF
( R2.hsGPA  <- summary( lm(hsGPA~ACT, data=gpa1) )$r.squared )
```

```
## [1] 0.1195815
```

Table 2.1: A table of the first eight columns and ten rows of the gpa1 data.

age	soph	junior	senior	senior5	male	campus	business
21	0	0	1	0	0	0	1
21	0	0	1	0	0	0	1
20	0	1	0	0	0	0	1
19	1	0	0	0	1	1	1
20	0	1	0	0	0	0	1
20	0	0	1	0	1	1	1
22	0	0	0	1	0	0	1
22	0	0	0	1	0	0	0
22	0	0	0	1	0	0	0
19	1	0	0	0	0	0	1

```
( VIF.hsGPA <- 1/(1-R2.hsGPA) )
```

```
## [1] 1.135823
```

The **car** package implements the command `vif()` for each regressor

```
vif(GPAres)
```

```
##    hsGPA      ACT
## 1.135823 1.135823
```

2.2.3 Multiple Regression Analysis: OLS Asymptotics

Note: Should we cover this? Most has been covered in SLR

2.2.4 Reporting Regression Results

As we start moving towards the comparing different regression models this section provides a discussion on how to report regression reports in `R`. Depending on your script (R scripts, R Markdown, bookdown) and what your desired output format is (LaTeX, word, html) the exact approach might differ. There are multiple packages to format regression or table output, most notabley *stargazer*[1], *huxtable* , *Hmisc* and *xtable*. One can also tidy the the regression output as well as tables with *broom* or *summarytool*. The wrapper `knitr::kable()` is a support function that renders the table in an R Markdown in a pretty way.

2.2.4.1 Table

```
knitr::kable(
  head(gpa1[,1:8], 10), booktabs = TRUE,
  caption = "A table of the first eight columns and ten rows of the gpa1 data."
)
```

Reporting summary statistics (transposed)

[1] Stargazer supports ton of options, including theming the LaTex output to journal styles. However, stargazer was written before R Markdown was really a thing, so it has excellent support for HTML and LaTeX output, but that's it. Including stargazer tables in an R Markdown document is a hassle. *huxtable* on the the contrary plays really nice with *broom* and the *tidyverse*. For more info see Andrew Heiss blog

```
descr(gpa1[,1:3], stats = c("mean", "sd", "min", "med", "max"), transpose = TRUE,
      omit.headings = TRUE, style = "rmarkdown")
```

Mean	Std.Dev	Min	Median	Max
20.9	1.27	19	21	30
0.0213	0.145	0	0	1
0.383	0.488	0	0	1

Including the `knitr::kable()` wrapper

```
knitr::kable(
  descr(gpa1[,1:3], stats = c("mean", "sd", "min", "med", "max"), transpose = TRUE,
        omit.headings = TRUE, style = "rmarkdown")
)
```

	Mean	Std.Dev	Min	Median	Max
age	20.8865248	1.2710637	19	21	30
soph	0.0212766	0.1448194	0	0	1
junior	0.3829787	0.4878462	0	0	1

```
model1 <- lm(colGPA ~ hsGPA , data = gpa1)
model2 <- lm(colGPA ~ hsGPA + ACT, data = gpa1)
model3 <- lm(colGPA ~ hsGPA + ACT + age, data = gpa1)
```

```
invisible(stargazer(
    list(model1,
         model2,
         model3)
    ,keep.stat = c("n", "rsq"), type = "latex", header = FALSE))# to have number of observations and R^
```

Table 2.2

	Dependent variable:		
	colGPA		
	(1)	(2)	(3)
hsGPA	0.482***	0.453***	0.482***
	(0.090)	(0.096)	(0.099)
ACT		0.009	0.009
		(0.011)	(0.011)
age			0.027
			(0.023)
Constant	1.415***	1.286***	0.618
	(0.307)	(0.341)	(0.663)
Observations	141	141	141
R^2	0.172	0.176	0.185
Note:	*p<0.1; **p<0.05; ***p<0.01		

```
stargazer(
    list(model1,
         model2,
         model3)
    ,keep.stat = c("n", "rsq"), type = "html", header = FALSE) # to have number of observations and R^2
```

Dependent variable:

colGPA

(1)

(2)

(3)

hsGPA

0.482***

0.453***

0.482***

(0.090)

(0.096)

(0.099)

ACT

0.009

0.009

(0.011)

(0.011)

age

0.027

(0.023)

Constant

1.415***

1.286***

0.618

(0.307)

(0.341)

(0.663)

Observations

141

141

141

R2

0.172

0.176

0.185

Note:

p<0.1; ***p<0.05;*** p<0.01

2.2.5 Model Formulae

2.2.5.1 Arithmetic operations within a formula

A model relating to birth weight to cigarette smoking of the mother during pregnancy and the family income.

```
data("bwght")
attach(bwght)
```

```
## The following object is masked _by_ .GlobalEnv:
##
##     bwght

## The following object is masked from gpa1 (pos = 3):
##
##     male

## The following objects are masked from bwght (pos = 14):
##
##     bwght, bwghtlbs, cigprice, cigs, cigtax, faminc, fatheduc,
##     lbwght, lfaminc, male, motheduc, packs, parity, white

## The following object is masked from gpa1 (pos = 15):
##
##     male

## The following object is masked from package:wooldridge:
##
##     bwght
```

```
lm1 <- lm(bwght ~ cigs + faminc, data = bwght)
# Weights in pounds, direct way
lm2 <- lm(I(bwght/16) ~ cigs + faminc, data = bwght)
# Packs of cigarettes
lm3 <- lm(bwght ~ I(cigs/20) + faminc, data = bwght)
```

See table **??**.

```
huxreg(lm1, lm2, lm3) %>%
    set_caption('(#tab:foo) Foo')
```

```
invisible(stargazer(
    list(lm1,
         lm2,
         lm3)
    ,keep.stat = c("n", "rsq"), type = "latex", header = FALSE))# to have number of observations and R^
```

Deviding the dependent variable by 16 changes all coefficients by the same facor $\frac{1}{16}$ and deviding the regressor by 20 changes its coefficients by the factor 20. Other statistics like R^2 are uneffected.

Table 2.3

	Dependent variable:		
	bwght	I(bwght/16)	bwght
	(1)	(2)	(3)
cigs	-0.463^{***}	-0.029^{***}	
	(0.092)	(0.006)	
I(cigs/20)			-9.268^{***}
			(1.832)
faminc	0.093^{***}	0.006^{***}	0.093^{***}
	(0.029)	(0.002)	(0.029)
Constant	116.974^{***}	7.311^{***}	116.974^{***}
	(1.049)	(0.066)	(1.049)
Observations	1,388	1,388	1,388
R^2	0.030	0.030	0.030
Note:			$^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$

2.2.5.2 Standardization: Beta coefficients

The standardized dependent variable y and regressor x_1 are

$$z_y = \frac{y - \bar{y}}{sd(y)} \tag{2.14}$$

and

$$z_{x1} = \frac{x_1 - \bar{x}_{x1}}{sd(x_1)} \tag{2.15}$$

They measure by how many *standard deviations* y changes as the respective indepdent variable increases by *one standard deviation.*

The model does not include a constant because all averages are removed in the standardization.

```r
data(hprice2)
```

```r
lm(scale(price)~0 +  scale(crime) +  scale(rooms) + scale(dist) +  scale(stratio), data = hprice2)
```

```
##
## Call:
## lm(formula = scale(price) ~ 0 + scale(crime) + scale(rooms) +
##     scale(dist) + scale(stratio), data = hprice2)
##
## Coefficients:
##   scale(crime)    scale(rooms)     scale(dist)  scale(stratio)
##      -0.191397        0.565694        0.003809       -0.246953
```

2.2.5.3 Logarithms, Quadratics and Polynomials

The model for houseprices as in Wooldrige:

$$log(price) = \beta_0 + \beta_1 log(nox) + \beta_2 log(dist) + \beta_3 rooms + \beta_4 rooms^2 + \beta_5 stratio + u \tag{2.16}$$

```
lm_hprice2 <- lm(log(price)~  log(nox) +  log(dist) + rooms + I(rooms^2) + stratio, data = hprice2)
summary(lm_hprice2)
```

```
##
## Call:
## lm(formula = log(price) ~ log(nox) + log(dist) + rooms + I(rooms^2) +
##     stratio, data = hprice2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.04285 -0.12774  0.02038  0.12650  1.25272
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.385477   0.566473  23.630  < 2e-16 ***
## log(nox)    -0.901682   0.114687  -7.862 2.34e-14 ***
## log(dist)   -0.086781   0.043281  -2.005  0.04549 *
## rooms       -0.545113   0.165454  -3.295  0.00106 **
## I(rooms^2)   0.062261   0.012805   4.862 1.56e-06 ***
## stratio     -0.047590   0.005854  -8.129 3.42e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2592 on 500 degrees of freedom
## Multiple R-squared:  0.6028, Adjusted R-squared:  0.5988
## F-statistic: 151.8 on 5 and 500 DF,  p-value: < 2.2e-16
```

- The quadratic term of rooms significantly positive coefficient $\hat{\beta}_4$ implying that the semi-elasticity increases with more rooms
- The negative coefficient for rooms indicates that for small number of rooms the price decreases and
- the positive coefficient for $rooms^2$ implies that for "large" value of rooms the price increases
- The number of rooms implying the smallest price can be found as

$$rooms^\star = \frac{-\beta_3}{2\beta_4} \approx 4.4 \tag{2.17}$$

```
beta3 <- lm_hprice2$coefficients[[4]]
beta4 <- lm_hprice2$coefficients[[5]]
-beta3 / (2 * beta4)
```

```
## [1] 4.37763
```

2.2.5.4 Interaction terms

Consider the following model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + u \tag{2.18}$$

where x_1, x_2, and Y are the same as before, but we have added a new interaction term x_1x_2 which multiplies x_1 and x_2, so we also have an additional β parameter β_3.

This model essentially creates two slopes and two intercepts, β_2 being the difference in intercepts and β_3 being the difference in slopes.

Recall that R reads **x1xx2** as $y\ x_1 + x_2 + x_1x_2$ and **x__1:x__2** as $y\ x_1x_2$.

```
data("attend")
```

```
# Estimate model with interaction effect:
(myres<-lm(stndfnl~atndrte*priGPA+ACT+I(priGPA^2)+I(ACT^2), data=attend))
```

```
##
## Call:
## lm(formula = stndfnl ~ atndrte * priGPA + ACT + I(priGPA^2) +
##     I(ACT^2), data = attend)
##
## Coefficients:
##    (Intercept)         atndrte          priGPA             ACT
##       2.050293       -0.006713       -1.628540       -0.128039
##    I(priGPA^2)        I(ACT^2)  atndrte:priGPA
##       0.295905        0.004533        0.005586
```

```
# Estimate for partial effect at priGPA=2.59:
b <- coef(myres)
b["atndrte"] + 2.59*b["atndrte:priGPA"]
```

```
##     atndrte
## 0.007754572
```

```
# Test partial effect for priGPA=2.59:
library(car)
linearHypothesis(myres,c("atndrte+2.59*atndrte:priGPA"))
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
674	519				
673	513	1	6.58	8.63	0.00341

2.2.6 MLR Prediction

```
data(gpa2)
```

```
# Regress and report coefficients
reg <- lm(colgpa~sat+hsperc+hsize+I(hsize^2),data=gpa2)
reg
```

```
##
## Call:
## lm(formula = colgpa ~ sat + hsperc + hsize + I(hsize^2), data = gpa2)
##
## Coefficients:
## (Intercept)          sat       hsperc        hsize   I(hsize^2)
##    1.492652     0.001492    -0.013856    -0.060881     0.005460
```

```r
# Generate data set containing the regressor values for predictions
cvalues <- data.frame(sat=1200, hsperc=30, hsize=5)

# Point estimate of prediction
predict(reg, cvalues)
```

```
##        1
## 2.700075
```

```r
# Point estimate and 95% confidence interval
predict(reg, cvalues, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 2.700075 2.661104 2.739047
```

```r
# Define three sets of regressor variables
cvalues <- data.frame(sat=c(1200,900,1400), hsperc=c(30,20,5),
                                                   hsize=c(5,3,1))
cvalues
```

sat	hsperc	hsize
1.2e+03	30	5
900	20	3
1.4e+03	5	1

```r
# Point estimates and 99% confidence intervals for these
predict(reg, cvalues, interval = "confidence", level=0.99)
```

```
##        fit      lwr      upr
## 1 2.700075 2.648850 2.751301
## 2 2.425282 2.388540 2.462025
## 3 3.457448 3.385572 3.529325
```

2.2.6.1 Prediction intervals

```r
# Regress (as before)
reg <- lm(colgpa~sat+hsperc+hsize+I(hsize^2),data=gpa2)

# Define three sets of regressor variables (as before)
cvalues <- data.frame(sat=c(1200,900,1400), hsperc=c(30,20,5),
                                                   hsize=c(5,3,1))

# Point estimates and 95% prediction intervals for these
predict(reg, cvalues, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 2.700075 1.601749 3.798402
## 2 2.425282 1.327292 3.523273
## 3 3.457448 2.358452 4.556444
```

Not covered

- 6.2.3 Effect Plots for Nonlinear Specification

2.3 MLR Analysis with Qualitative Regressors

2.3.1 Dummy variabes

```
data(wage1)
```

```
lm1_wage1 <- lm(wage ~ female+educ+exper+tenure, data=wage1)
summary(lm1_wage1)
```

```
##
## Call:
## lm(formula = wage ~ female + educ + exper + tenure, data = wage1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7675 -1.8080 -0.4229  1.0467 14.0075
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.56794    0.72455  -2.164   0.0309 *
## female      -1.81085    0.26483  -6.838 2.26e-11 ***
## educ         0.57150    0.04934  11.584  < 2e-16 ***
## exper        0.02540    0.01157   2.195   0.0286 *
## tenure       0.14101    0.02116   6.663 6.83e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.958 on 521 degrees of freedom
## Multiple R-squared:  0.3635, Adjusted R-squared:  0.3587
## F-statistic:  74.4 on 4 and 521 DF,  p-value: < 2.2e-16
```

On average a woman makes $ 0 per less than a man with the *same* education, experience, and tenure.

```
lm2_wage1 <- lm(log(wage)~married*female+educ+exper+I(exper^2)+tenure+I(tenure^2), data=wage1)
summary(lm2_wage1)
```

```
##
## Call:
## lm(formula = log(wage) ~ married * female + educ + exper + I(exper^2) +
##     tenure + I(tenure^2), data = wage1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.89697 -0.24060 -0.02689  0.23144  1.09197
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.3213781  0.1000090   3.213 0.001393 **
## married         0.2126757  0.0553572   3.842 0.000137 ***
## female         -0.1103502  0.0557421  -1.980 0.048272 *
## educ            0.0789103  0.0066945  11.787  < 2e-16 ***
```

```
## exper           0.0268006  0.0052428   5.112 4.50e-07 ***
## I(exper^2)     -0.0005352  0.0001104  -4.847 1.66e-06 ***
## tenure          0.0290875  0.0067620   4.302 2.03e-05 ***
## I(tenure^2)    -0.0005331  0.0002312  -2.306 0.021531 *
## married:female -0.3005931  0.0717669  -4.188 3.30e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3933 on 517 degrees of freedom
## Multiple R-squared:  0.4609, Adjusted R-squared:  0.4525
## F-statistic: 55.25 on 8 and 517 DF,  p-value: < 2.2e-16
```

Your turn

1. What is the refernce group in this model?
2. Ceteris paribus, how much more wage do single males make relative to the reference group?
3. Ceteris paribus, how much more wage do single females make relative to the reference group?
4. Ceteris paribus, how much less do married females make than single females?
5. Do the results make sense economically. What socio-economic factors could explain the results?

```
df_lm2_wage1 <- tidy(lm2_wage1)
# Singe male
marriedmale <- df_lm2_wage1 %>%
  filter(term == "married") %>%
  dplyr::select(estimate) %>%
  pull() # pull out the single coefficient value of the dataframe
# Single female
singlefemale <- df_lm2_wage1 %>%
  filter(term == "female") %>%
  dplyr::select(estimate) %>%
  pull() # pull out the single coefficient value of the dataframe
marriedfemale <- df_lm2_wage1 %>%
  filter(term == "married:female") %>%
  dplyr::select(estimate) %>%
  pull() # pull out the single coefficient value of the dataframe
married<- df_lm2_wage1 %>%
  filter(term == "married") %>% #
  dplyr::select(estimate) %>%
  pull() # pull out the single coefficient value of the dataframe
```

A:

1. Reference group: *single* and *male*
2. Cp. married males make 21.3% more than single males.
3. Cp. a single female makes -11.0% less than the reference group.
4. Married females make 8.79% less than single females.
5. There seems to be a marriage premium for men but for women the marriage premium is negative.

2.3.2 Logical variables

```
# replace "female" with logical variable
wage1$female <- as.logical(wage1$female)
```

```r
table(wage1$female)
```

```
##
## FALSE  TRUE
##   274   252
```

```r
# regression with logical variable
lm(wage ~ female+educ+exper+tenure, data=wage1)
```

```
##
## Call:
## lm(formula = wage ~ female + educ + exper + tenure, data = wage1)
##
## Coefficients:
## (Intercept)   femaleTRUE         educ        exper       tenure
##     -1.5679      -1.8109       0.5715       0.0254       0.1410
```

2.3.3 Factor variables

As discussed in the R introduction, categorial variables encoded as factors are special *animals* in R. They are immensely useful in a regression when you have a categorical variable with many levels (e.g. "Very Bad", "Bad", "Good", "Very Good") but can create a set of subtile issues. Here, we discuss the base R way and the more robust tidyverse way of dealing with factors in the area of regression modelling.

Factor variables can be directly added to the list of regressors. R is clever enough to implicitly add $g-1$ dummy variables if the factor has g outcomes.

```r
data(CPS1985,package="AER")
str(CPS1985)
```

```
## 'data.frame':    534 obs. of  11 variables:
##  $ wage      : num  5.1 4.95 6.67 4 7.5 ...
##  $ education : num  8 9 12 12 12 13 10 12 16 12 ...
##  $ experience: num  21 42 1 4 17 9 27 9 11 9 ...
##  $ age       : num  35 57 19 22 35 28 43 27 33 27 ...
##  $ ethnicity : Factor w/ 3 levels "cauc","hispanic",..: 2 1 1 1 1 1 1 1 1 1 ...
##  $ region    : Factor w/ 2 levels "south","other": 2 2 2 2 2 2 1 2 2 2 ...
##  $ gender    : Factor w/ 2 levels "male","female": 2 2 1 1 1 1 1 1 1 1 ...
##  $ occupation: Factor w/ 6 levels "worker","technical",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ sector    : Factor w/ 3 levels "manufacturing",..: 1 1 1 3 3 3 3 3 1 3 ...
##  $ union     : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 1 1 1 1 ...
##  $ married   : Factor w/ 2 levels "no","yes": 2 2 1 1 2 1 1 1 2 1 ...
```

```r
# Table of categories and frequencies for two factor variables:
table(CPS1985$gender)
```

```
##
##   male female
##    289    245
```

```r
table(CPS1985$occupation)
```

```
##
##     worker  technical   services     office      sales management
##        156        105         83         97         38         55
```

```
levels(CPS1985$occupation)
```

```
## [1] "worker"     "technical"  "services"   "office"     "sales"
## [6] "management"
```

```
levels(CPS1985$gender)
```

```
## [1] "male"   "female"
```

```
# Directly using factor variables in regression formula:
lm(log(wage) ~ education+experience+gender+occupation, data=CPS1985)
```

```
##
## Call:
## lm(formula = log(wage) ~ education + experience + gender + occupation,
##     data = CPS1985)
##
## Coefficients:
##          (Intercept)             education            experience
##              0.97629               0.07586               0.01188
##         genderfemale   occupationtechnical    occupationservices
##             -0.22385               0.14246              -0.21004
##     occupationoffice       occupationsales  occupationmanagement
##             -0.05477              -0.20757               0.15254
```

```
# Fragile method (base R)
# Manually redefine the  reference category:
CPS1985$gender <- relevel(CPS1985$gender,"female")
CPS1985$occupation <- relevel(CPS1985$occupation,"management")

# Rerun regression:
lm(log(wage) ~ education+experience+gender+occupation, data=CPS1985)
```

```
##
## Call:
## lm(formula = log(wage) ~ education + experience + gender + occupation,
##     data = CPS1985)
##
## Coefficients:
##         (Intercept)            education           experience
##             0.90498              0.07586              0.01188
##          gendermale     occupationworker  occupationtechnical
##             0.22385             -0.15254             -0.01009
##  occupationservices     occupationoffice      occupationsales
##            -0.36259             -0.20731             -0.36011
```

```
# Robust method (tidyverse)
# Manually redefine the  reference category (back to default):
CPS1985 <- CPS1985 %>%
  mutate(gender = fct_relevel(gender, "female")) %>%
  mutate(occupation = fct_relevel(occupation, "worker"))

lm(log(wage) ~ education+experience+gender+occupation, data=CPS1985)
```

```
##
## Call:
## lm(formula = log(wage) ~ education + experience + gender + occupation,
```

```
##     data = CPS1985)
##
## Coefficients:
##          (Intercept)              education             experience
##              0.75244                0.07586                0.01188
##           gendermale  occupationmanagement   occupationtechnical
##              0.22385                0.15254                0.14246
##   occupationservices      occupationoffice       occupationsales
##             -0.21004               -0.05477               -0.20757
```

2.3.3.1 Breaking a numeric variable into categories

```
data(lawsch85)
str(lawsch85$rank)
```

```
##  int [1:156] 128 104 34 49 95 98 124 157 145 91 ...
```

```
# Define cut points for the rank
cutpts <- c(0,10,25,40,60,100,175)

# Create factor variable containing ranges for the rank
lawsch85$rankcat <- cut(lawsch85$rank, cutpts)

# Display frequencies
table(lawsch85$rankcat)
```

```
##
##    (0,10]   (10,25]   (25,40]   (40,60]  (60,100] (100,175]
##        10        16        13        18        37        62
```

```
# Choose reference category
lawsch85$rankcat <- relevel(lawsch85$rankcat,"(100,175]")

# Run regression
(res <- lm(log(salary)~rankcat+LSAT+GPA+log(libvol)+log(cost), data=lawsch85))
```

```
##
## Call:
## lm(formula = log(salary) ~ rankcat + LSAT + GPA + log(libvol) +
##     log(cost), data = lawsch85)
##
## Coefficients:
##     (Intercept)    rankcat(0,10]   rankcat(10,25]   rankcat(25,40]
##       9.1652952        0.6995659        0.5935434        0.3750763
##  rankcat(40,60]  rankcat(60,100]             LSAT              GPA
##       0.2628191        0.1315950        0.0056908        0.0137255
##     log(libvol)        log(cost)
##       0.0363619        0.0008412
```

```
# ANOVA table
car::Anova(res)
```

The regression results imply that graduates from the top 100 schools collect a starting salary which is around 70% higher than those of the schools below rank 100. This approximation is inaccurate with these large numbers and the coefficient of 0.7 actually implies a difference of ex(0.7-1) = 1.103 or 101.3%.

Sum Sq	Df	F value	Pr(>F)
1.87	5	51	1.17e-28
0.0253	1	3.45	0.0655
0.000251	1	0.0342	0.854
0.0143	1	1.95	0.165
8.21e-06	1	0.00112	0.973
0.924	126		

2.3.4 Interactions and differences in regression functions across groups

Dummy variables and factor variables can be interacted just like any other variable

- Use the **subset** option of `lm` to directly define the estimation sample
- The dummy variable *female* is interacted with all other regressor
- The F test for all interaction effects is performed using the function `linearHypothesis` from the **car** package

```
data(gpa3)

# Model with full interactions with female dummy (only for spring data)
reg<-lm(cumgpa~female*(sat+hsperc+tothrs), data=gpa3, subset=(spring==1))
summary(reg)
```

```
##
## Call:
## lm(formula = cumgpa ~ female * (sat + hsperc + tothrs), data = gpa3,
##     subset = (spring == 1))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.51370 -0.28645 -0.02306  0.27555  1.24760
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.4808117  0.2073336   7.142 5.17e-12 ***
## female        -0.3534862  0.4105293  -0.861  0.38979
## sat            0.0010516  0.0001811   5.807 1.40e-08 ***
## hsperc        -0.0084516  0.0013704  -6.167 1.88e-09 ***
## tothrs         0.0023441  0.0008624   2.718  0.00688 **
## female:sat     0.0007506  0.0003852   1.949  0.05211 .
## female:hsperc -0.0005498  0.0031617  -0.174  0.86206
## female:tothrs -0.0001158  0.0016277  -0.071  0.94331
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4678 on 358 degrees of freedom
## Multiple R-squared:  0.4059, Adjusted R-squared:  0.3943
## F-statistic: 34.95 on 7 and 358 DF,  p-value: < 2.2e-16
```

```
# F-Test from package "car". H0: the interaction coefficients are zero
# matchCoefs(...) selects all coeffs with names containing "female"
```

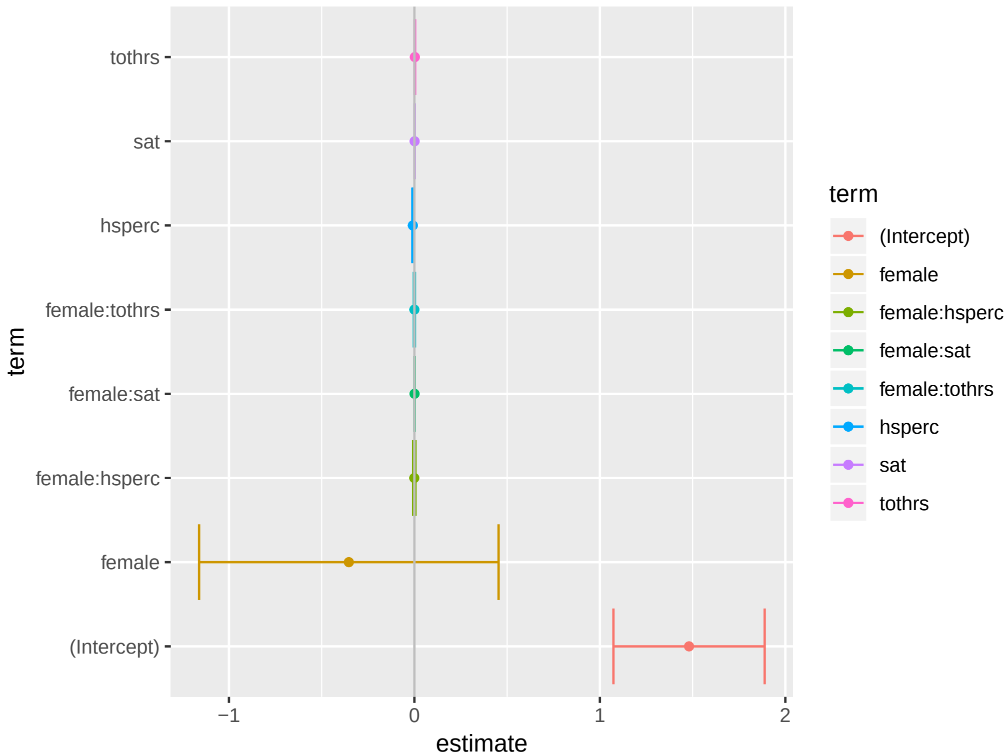


Figure 2.21: Coefficient plots

```r
linearHypothesis(reg, matchCoefs(reg, "female"))
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
362	85.5				
358	78.4	4	7.16	8.18	2.54e-06

2.3.4.1 Visualizing coefficients

```r
treg <- tidy(reg, conf.int = TRUE)
```

```r
ggplot(treg, aes(estimate, term, color = term)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))  +
    geom_vline(xintercept = 0, color = "grey")
```

2.4 Heteroskedasticity

The homoskedasticity assumptions SLR.5 and MLR.5 require that the variance of the error term is unrelated to the regressors, i.e.

$$Var(u|x_1, \ldots, x_n) = \sigma^2 \tag{2.19}$$

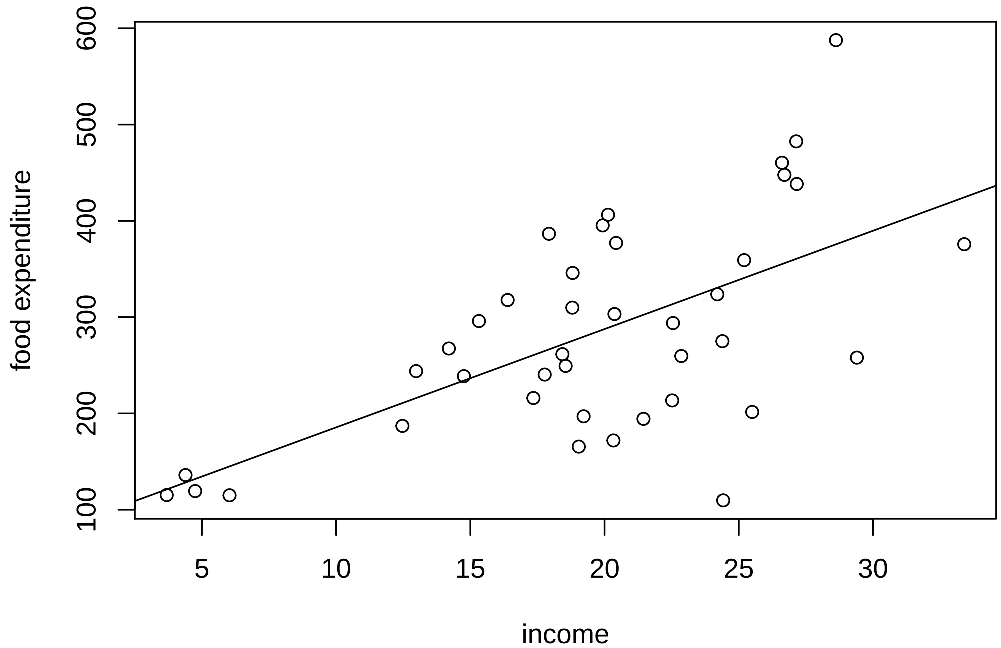


Figure 2.22: Heteroskedasticity in the 'food' data

Unbiasedness and consistency do not depend on this assumption, but the sampling distribution does. If homoskedasticity is violated, the standard errors are invalid and all inferencences from t, F, and other tests based on them are unreliable.

There are various ways of dealing with heteroskedasticity in R. The **car** package provides linear hypothesis. For high-dimensional fixed effects the **lfe** package is a good alternative. It also allows to specify clusters as part of the formula. A good balance between functionality and ease of use is provided by the **sandwich** package **?**.

2.4.1 Spotting Heteroskedasticity in Scatter Plots

```
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.

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

```
plot(yhat,res, xlab="fitted values", ylab="residuals")
```

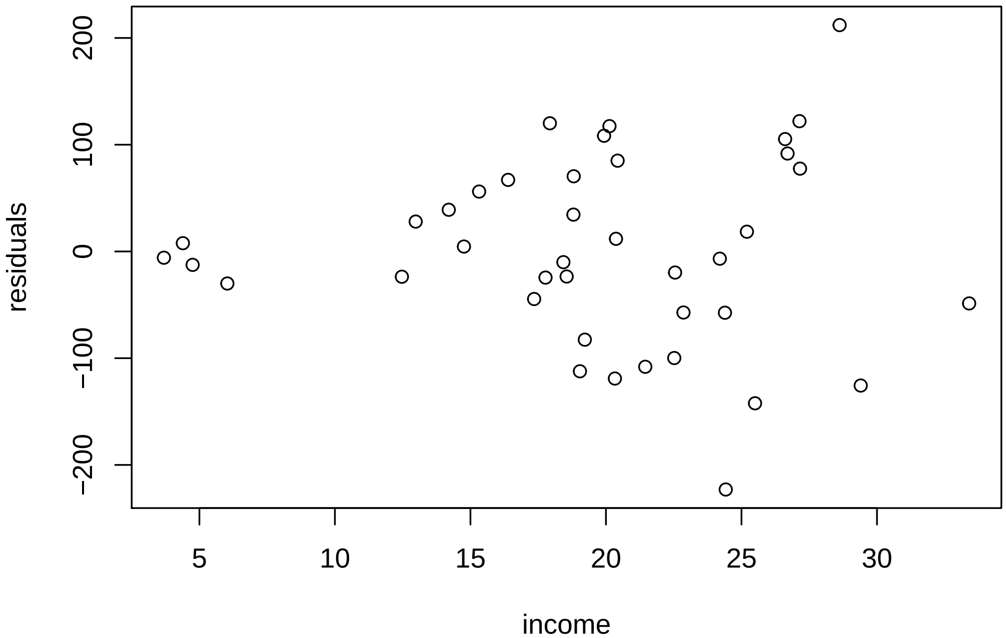


Figure 2.23: Residual plots in the 'food' model

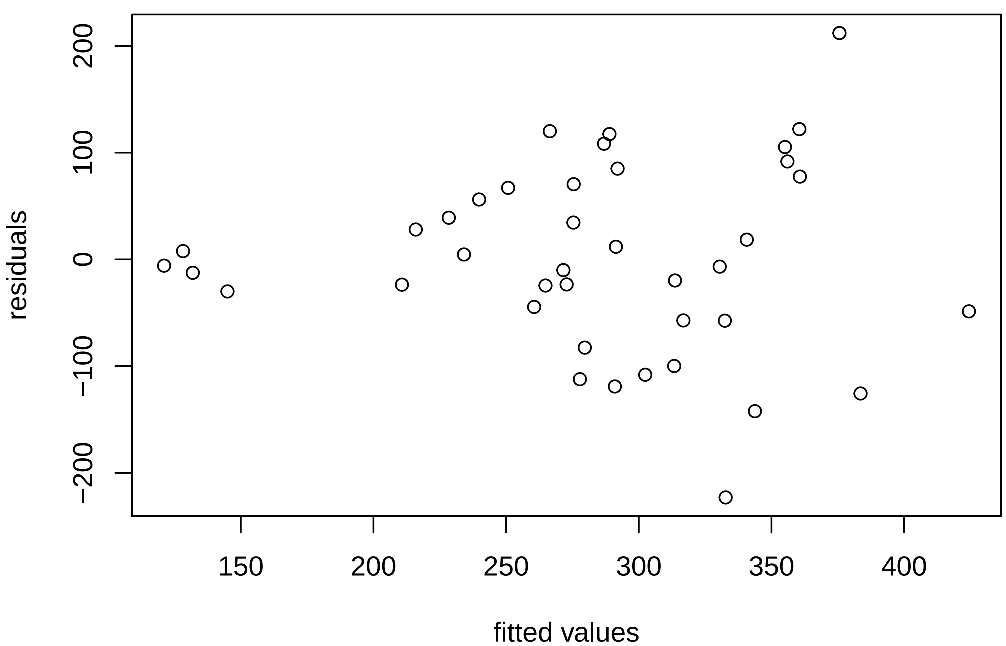


Figure 2.24: Residual plots in the 'food' model

2.4.2 Heteroskedasticity Tests

```
data(gpa3, package='wooldridge')


# Estimate model (only for spring data)
reg <- lm(cumgpa~sat+hsperc+tothrs+female+black+white,
                                     data=gpa3, subset=(spring==1))


# Breusch-Pagan (BP) Test

bptest(reg)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  reg
## BP = 44.557, df = 6, p-value = 5.732e-08
```

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.

```
# Usual SE:
coeftest(reg)
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  1.47006477  0.22980308  6.3971 4.942e-10 ***
## sat          0.00114073  0.00017856  6.3885 5.197e-10 ***
## hsperc      -0.00856636  0.00124042 -6.9060 2.275e-11 ***
## tothrs       0.00250400  0.00073099  3.4255 0.0006847 ***
## female       0.30343329  0.05902033  5.1412 4.497e-07 ***
## black       -0.12828368  0.14737012 -0.8705 0.3846164
## white       -0.05872173  0.14098956 -0.4165 0.6772953
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Refined White heteroscedasticity-robust SE:
coeftest(reg, vcov=hccm)
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  1.47006477  0.22938036  6.4089 4.611e-10 ***
## sat          0.00114073  0.00019532  5.8402 1.169e-08 ***
## hsperc      -0.00856636  0.00144359 -5.9341 6.963e-09 ***
## tothrs       0.00250400  0.00074930  3.3418   0.00092 ***
## female       0.30343329  0.06003964  5.0539 6.911e-07 ***
## black       -0.12828368  0.12818828 -1.0007   0.31762
```

```
## white       -0.05872173  0.12043522 -0.4876   0.62615
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
cov3 <- hccm(reg, type="hc3") # hc3 is the standard method
ref.HC3 <- coeftest(reg, vcov.=cov3)

# Supply other White corrections
cov1 <- hccm(reg, type="hc1")
ref.HC1 <- coeftest(reg, vcov.=cov1)
```

Another way of dealing with heteroskedasticity is to use the lmrob() function from the {robustbase} package[2]. This package is quite interesting, and offers quite a lot of functions for robust linear, and nonlinear, regression models. Running a robust linear regression is just the same as with lm():

```
library(robustbase)
regrobfit <- lmrob(cumgpa~sat+hsperc+tothrs+female+black+white,
                                      data=gpa3, subset=(spring==1))

summary(regrobfit)
```

```
##
## Call:
## lmrob(formula = cumgpa ~ sat + hsperc + tothrs + female + black + white,
##     data = gpa3, subset = (spring == 1))
##  \--> method = "MM"
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.57535 -0.30124 -0.02834  0.26687  1.27950
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.4693758  0.2315018   6.347 6.62e-10 ***
## sat          0.0011185  0.0001953   5.727 2.17e-08 ***
## hsperc      -0.0079056  0.0014293  -5.531 6.14e-08 ***
## tothrs       0.0021841  0.0007750   2.818   0.0051 **
## female       0.3002542  0.0599150   5.011 8.50e-07 ***
## black       -0.1281927  0.1268974  -1.010   0.3131
## white       -0.0305168  0.1181863  -0.258   0.7964
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Robust residual standard error: 0.4201
## Multiple R-squared:  0.411,  Adjusted R-squared:  0.4012
## Convergence in 15 IRWLS iterations
##
## Robustness weights:
##  22 weights are ~= 1. The remaining 344 ones are summarized as
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1291  0.8670  0.9471  0.8933  0.9854  0.9987
## Algorithmic parameters:
##        tuning.chi                bb        tuning.psi        refine.tol
##         1.548e+00         5.000e-01         4.685e+00         1.000e-07
##           rel.tol         scale.tol         solve.tol       eps.outlier
```

[2] This example has been adapted from the blogpost of **?**