MEM52220 - Applied Econometrics

Nicolas Reigl

2018-08-20

Contents

Introduction

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

Prerequisites

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

In order to reproduce the examples in this script You need the statistical software package R. Additionally we recommend using the RStudio integrated developer environment (IDE) which will improve Your R working experience.

Installation of R and RStudio

Go to the following webpages and download the software adequate for Your operating system:

1. The Statistical Software Package R: http://cran.r-project.org
2. The GUI RStudio: http://www.rstudio.com/products/rstudio/download/

Install R first and then procede with RStudio. Afterwards start RStudio.

Resources

Our primary resource is **?** [1]. The book is availabe as free online version. For theoretical concepts I refer to **?**[2].

Standing on the shoulders of giants

This lecture material would not possible without many other open-source econometrics teaching materials and of course the R package developers. In addition to the main resources, examples and code of this workbook have been drawn from a number of free econometrics ebooks, blogs, R-vignette help pages and other researchers teaching materials.

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

[2] Introductory Econometrics: A Modern Approach.

Attribution

Chapter 1 Introduction to R is copied almost entirely from appliedstats by David Dalpiaz. I added a couple of practical tasks and made some minor edits. Chapter 2 is partly based on appliedstats, but only up to scatterplots.

- Using R for Introduction to Econometrics
- Applied Statistics with R, **?**
- Principles of Econometrics with R
- Introduction to Econometrics with R, **?**

Software information and conventions

The R session information when compiling this book is shown below:

```
sessionInfo()
```

```
## R version 3.5.1 (2018-07-02)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] censReg_0.5-26       nnet_7.3-12          prediction_0.3.6
##  [4] margins_0.3.23       sampleSelection_1.2-0 maxLik_1.3-4
##  [7] miscTools_0.6-22     MCMCpack_1.4-3       MASS_7.3-50
## [10] coda_0.19-1          bindrcpp_0.2.2       modelr_0.1.2
## [13] quantreg_5.36        SparseM_1.77         robustbase_0.93-2
## [16] OutliersO3_0.5.4     scales_1.0.0         summarytools_0.8.7
## [19] PoEdata_0.1.0        AER_1.2-5            survival_2.42-6
## [22] sandwich_2.4-0       lmtest_0.9-36        zoo_1.8-3
## [25] stargazer_5.2.2      VIM_4.7.0            data.table_1.11.4
## [28] colorspace_1.3-2     mice_3.3.0           lattice_0.20-35
## [31] huxtable_4.1.0       knitr_1.20           car_3.0-0
## [34] carData_3.0-1        plot3D_1.1.1         ggfortify_0.4.5
## [37] ggpubr_0.1.7         magrittr_1.5         broom_0.5.0
## [40] forcats_0.3.0        stringr_1.3.1        dplyr_0.7.6
## [43] purrr_0.2.5          readr_1.1.1          tidyr_0.8.1
## [46] tibble_1.4.2         tidyverse_1.2.1.9000 wooldridge_1.3.0
## [49] ggplot2_3.0.0.9000
##
## loaded via a namespace (and not attached):
##   [1] readxl_1.1.0         backports_1.1.2      cellWise_2.0.10
##   [4] VGAM_1.0-5           plyr_1.8.4           lazyeval_0.2.1.9000
##   [7] sp_1.3-1            splines_3.5.1       pryr_0.1.4
##  [10] digest_0.6.15       htmltools_0.3.6     cluster_2.0.7-1
##  [13] openxlsx_4.1.0      robustX_1.2-2       matrixStats_0.54.0
##  [16] bdsmatrix_1.3-3     rvest_0.3.2         haven_1.1.2
##  [19] pan_1.6             xfun_0.3            crayon_1.3.4
##  [22] RCurl_1.95-4.11     jsonlite_1.5        FastPCS_0.1.3
##  [25] lme4_1.1-17         bindr_0.1.1         glue_1.3.0
##  [28] gtable_0.2.0        MatrixModels_0.4-1  DEoptimR_1.0-8
##  [31] jomo_2.6-3          abind_1.4-5         rapportools_1.0
##  [34] mvtnorm_1.0-8       GGally_1.4.0        Rcpp_0.12.18
##  [37] laeken_0.4.6        flashClust_1.01-2   foreign_0.8-71
##  [40] mclust_5.4.1        Formula_1.2-3       stats4_3.5.1
##  [43] vcd_1.4-4           httr_1.3.1          FNN_1.1.2.1
##  [46] RColorBrewer_1.1-2  pkgconfig_2.0.1     reshape_0.8.7
##  [49] plm_1.6-6           systemfit_1.1-22    tidyselect_0.2.4
##  [52] labeling_0.3        rlang_0.2.1         reshape2_1.4.3
##  [55] munsell_0.5.0       cellranger_1.1.0    tools_3.5.1
##  [58] cli_1.0.0           HDoutliers_1.0      glmmML_1.0.3
##  [61] evaluate_0.11       yaml_2.2.0          mcmc_0.9-5
##  [64] zip_1.0.0           pander_0.6.2        mitml_0.3-6
##  [67] nlme_3.1-137        leaps_3.0           xml2_1.2.0
##  [70] compiler_3.5.1      rstudioapi_0.7      curl_3.2
##  [73] e1071_1.7-0         stringi_1.2.4       highr_0.7
##  [76] Matrix_1.2-14       nloptr_1.0.4        pillar_1.3.0
##  [79] cowplot_0.9.3       bitops_1.0-6        R6_2.2.2
##  [82] bookdown_0.7.15     gridExtra_2.3       rio_0.5.10
##  [85] codetools_0.2-15    boot_1.3-20         assertthat_0.2.0
##  [88] rprojroot_1.3-2     withr_2.1.2         rlist_0.4.6.1
##  [91] parallel_3.5.1      hms_0.4.2.9000      rpart_4.1-13
##  [94] class_7.3-14        minqa_1.2.4         rmarkdown_1.10
##  [97] misc3d_0.8-4        scatterplot3d_0.3-41 lubridate_1.7.4
## [100] FactoMineR_1.41
```

I do not add prompts (`>` and `+`) to R source code in this book, and I comment out the text output with two hashes `##` by default, as you can see from the R session information above. This is for your convenience when you want to copy and run the code (the text output will be ignored since it is commented out). Package names are in bold text (e.g., **rmarkdown**), and inline code and filenames are formatted in a typewriter font (e.g., `knitr::knit('foo.Rmd')`). Function names are followed by parentheses (e.g., `bookdown::render_book()`), function arguments are given without parentheses (e.g. `config_file =`). The double-colon operator `::` means accessing an object from a package.

Acknowledgements

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

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
            "mice",  # multiple imputation
            "VIM", # visualizing missing data
            "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
            "scales", # scale helper functions such as percent
            "OutliersO3", # Outlier comparison method
            "robustbase", # Basic robust statistics
            "quantreg", # Quantile regression
            "modelr", # model simulation/ boostraping the modern way
            "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)
```

Chapter 1

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
```

1.1 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.
- R data sets for "Principles of Econometrics" by **?**
- Build in examples such as the `airquality` dataset

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 $\beta < 0$

Table 1.1: A table of the first eight columns and ten rows of the ceosal1 data.

salary	pcsalary	sales	roe	pcroe	ros	indus	finance
1095	20	27595.0	14.1	106.4	191	1	0
1001	32	9958.0	10.9	-30.6	13	1	0
1122	9	6125.9	23.5	-16.3	14	1	0
578	-9	16246.0	5.9	-25.7	-21	1	0
1368	7	21783.2	13.8	-3.0	56	1	0
1145	5	6021.4	20.0	1.0	55	1	0
1078	10	2266.7	16.4	-5.9	62	1	0
1094	7	2966.8	16.3	-1.6	44	1	0
1237	16	4570.2	10.5	-70.2	37	1	0
833	5	2830.0	26.3	-23.9	37	1	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

```r
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.

```r
View(ceosal1) # For the compilation we omit the full View()
```

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

```r
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
```

```
nrow(ceosal1)
```

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

```
ncol(ceosal1)
```

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

```
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 **?**.

```
# 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.

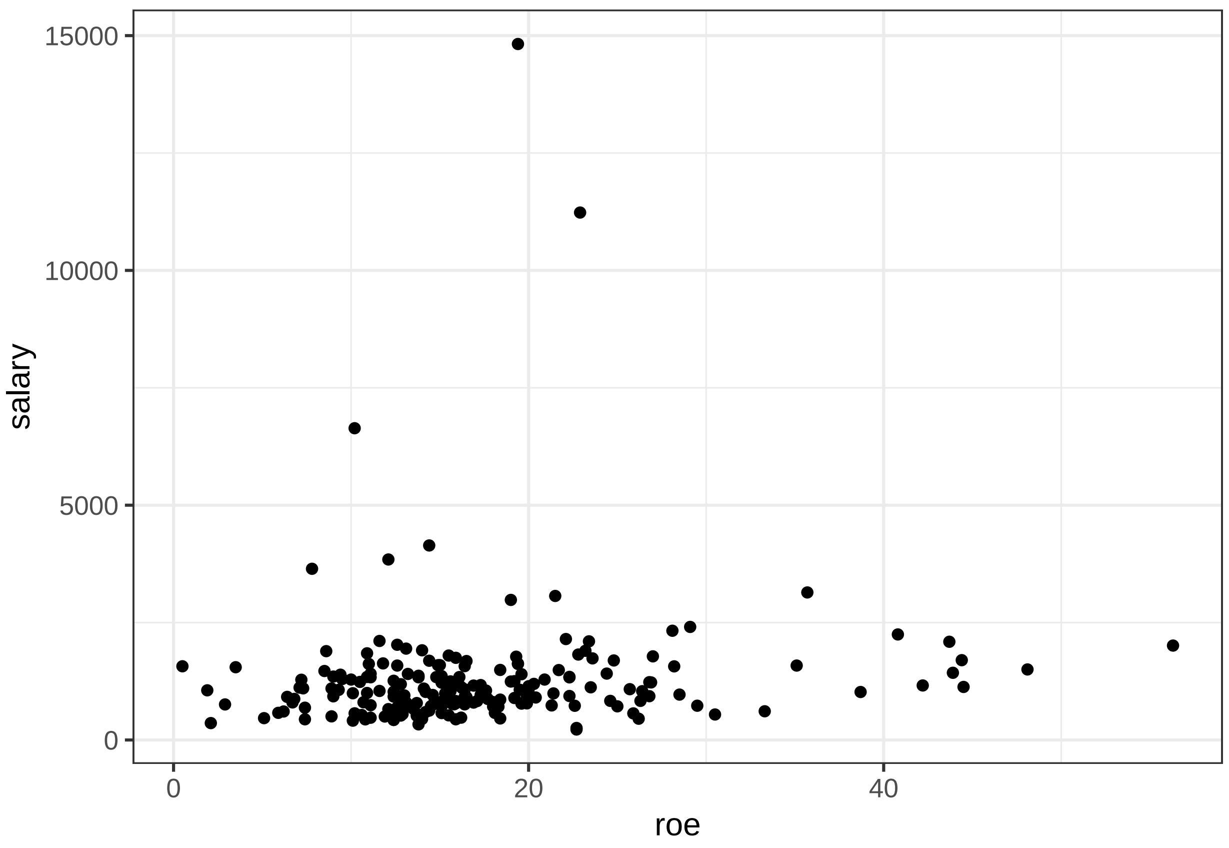


Figure 1.1: Relationship between ROE and Salary

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{1.1}$$

The ordinary least squares (OLS) estimators are

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x} \tag{1.2}$$

Ingredients for the OLS formulas

```
attach(ceosal1)
```

```
## The following objects are masked from ceosal1 (pos = 18):
##
##     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
```

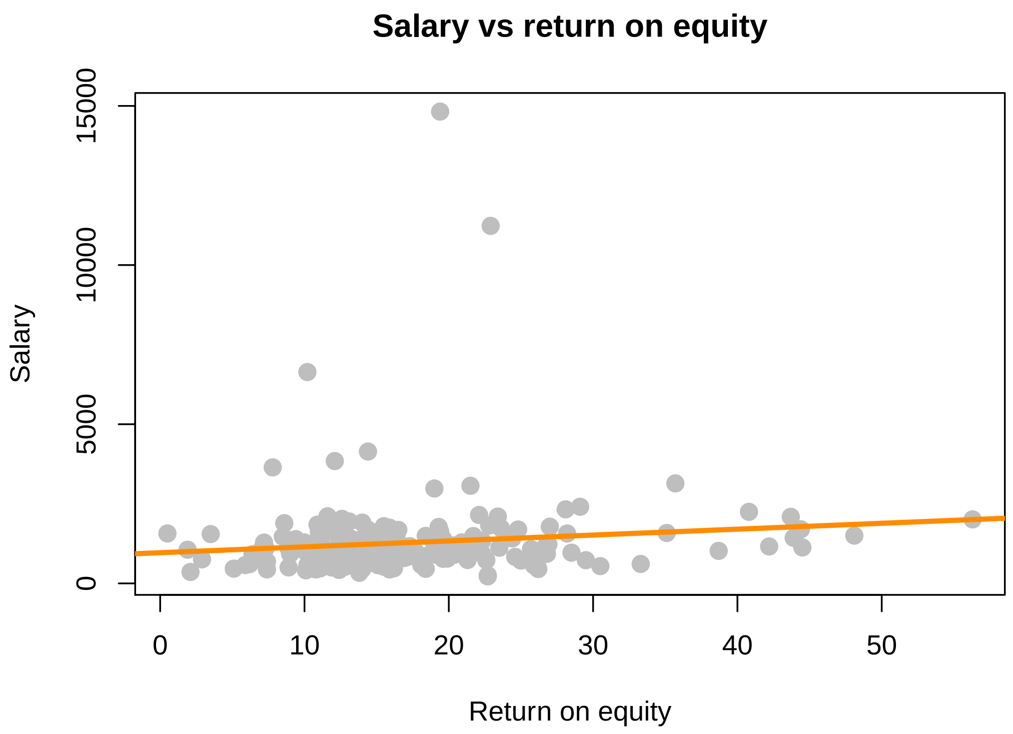


Figure 1.2: OLS regression base Rstyle

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() +
  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
```

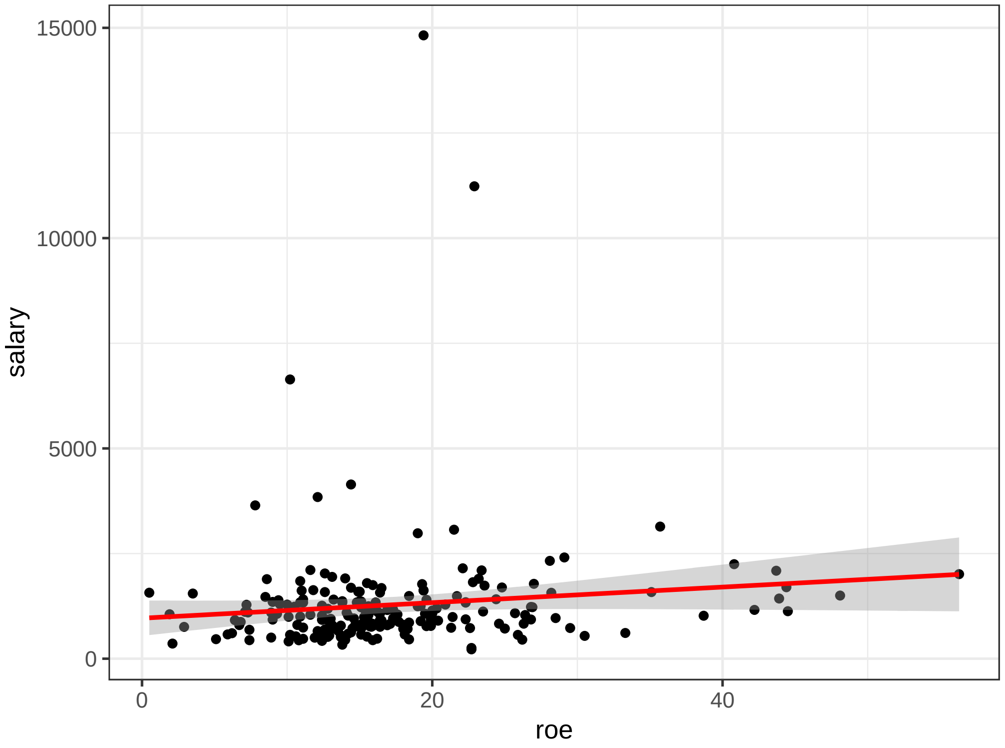


Figure 1.3: OLS regression ggplot2 style

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:
## 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.

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

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

Your turn

Recall that the residual sum of squares (SSR) is

$$R^2 = \frac{Var(\hat{y})}{Var(y)} = 1 - \frac{Var(\hat{u})}{Var(y)} \tag{1.3}$$

Calculate R^2 manually:

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

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

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

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

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

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

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

```
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

```
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."

1.1.1 Regression through the Origin and Regression on a Constant

Regression without intercept (through origin)

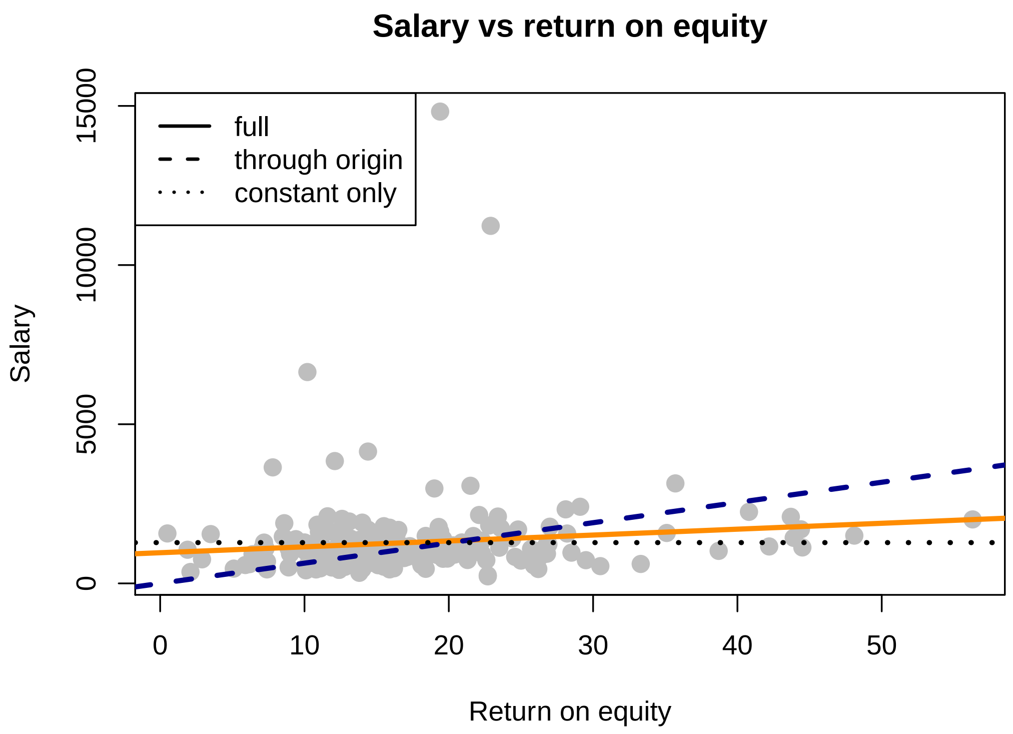


Figure 1.4: Regression through the Origin and on a Constant

```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")
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)
```

1.1.2 Simulating SLR

1.1.2.0.1 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: $x_1, \dots, x_n$
- 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

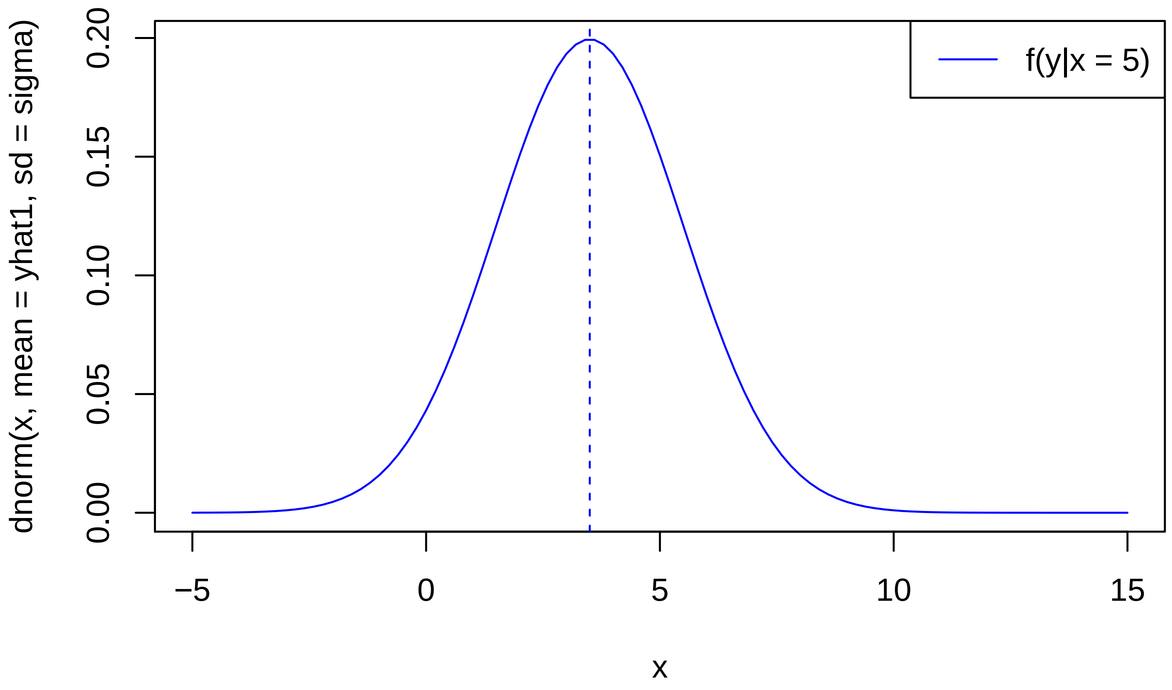
```
b0<- 1
b1 <- 0.5
sigma <- 2 # standard deviation of the error term u
x1 <- 5
```

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.

$$var(b_2) = \frac{\sigma^2}{\sum (x_1 - \bar{x})} \tag{1.4}$$

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

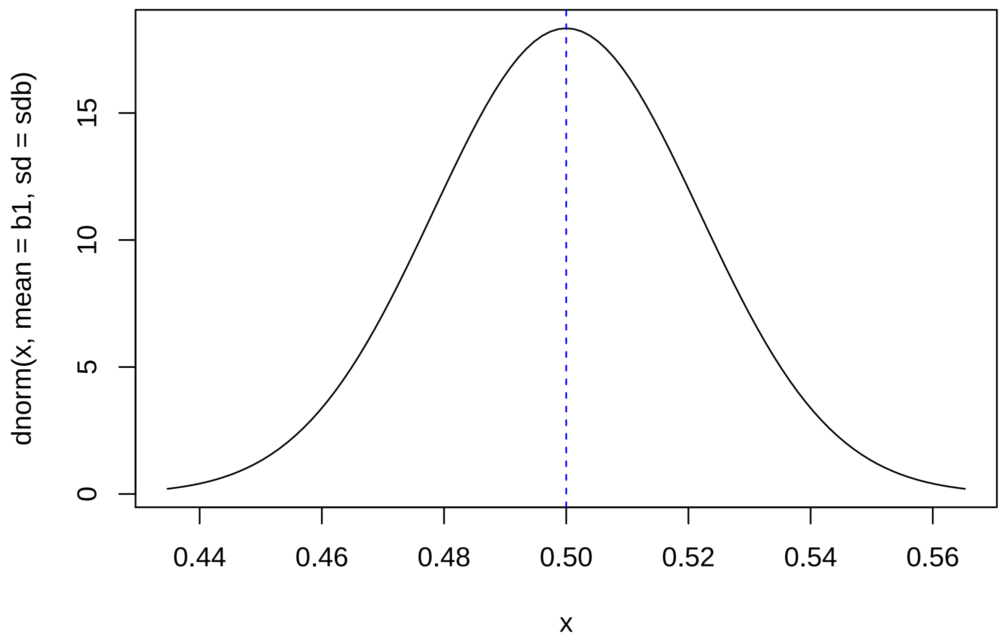


Figure 1.5: The theoretical (true) probability density function of b1

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

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

```r
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)
```

```r
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"
  )
```

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.

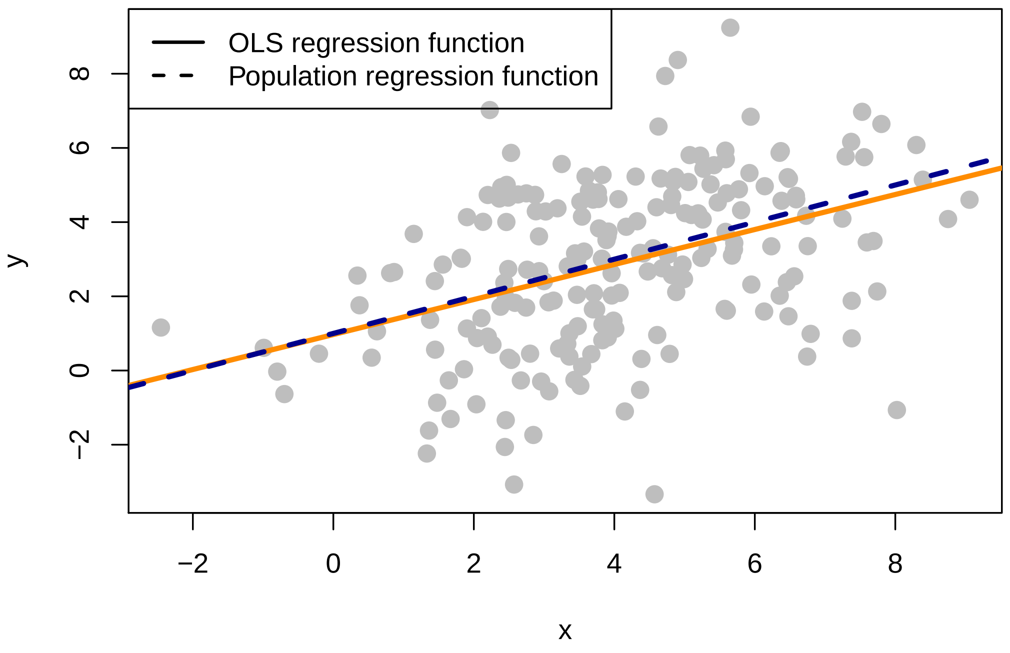


Figure 1.6: Simulated Sample and OLS Regression Line

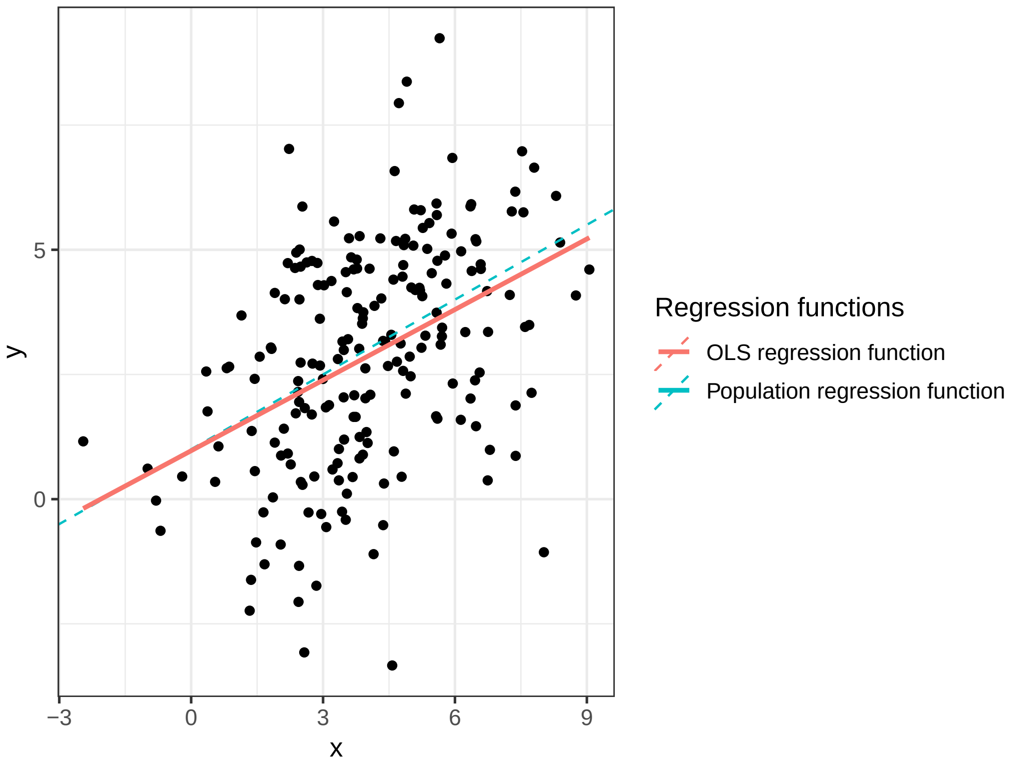


Figure 1.7: Simulated Sample and OLS Regression Line (gpplot Style)

```r
# 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<-1.0; 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"]
}
```

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

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

```r
mean(b1hat)
```

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

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

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

```r
var(b1hat)
```

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

```r
# 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.

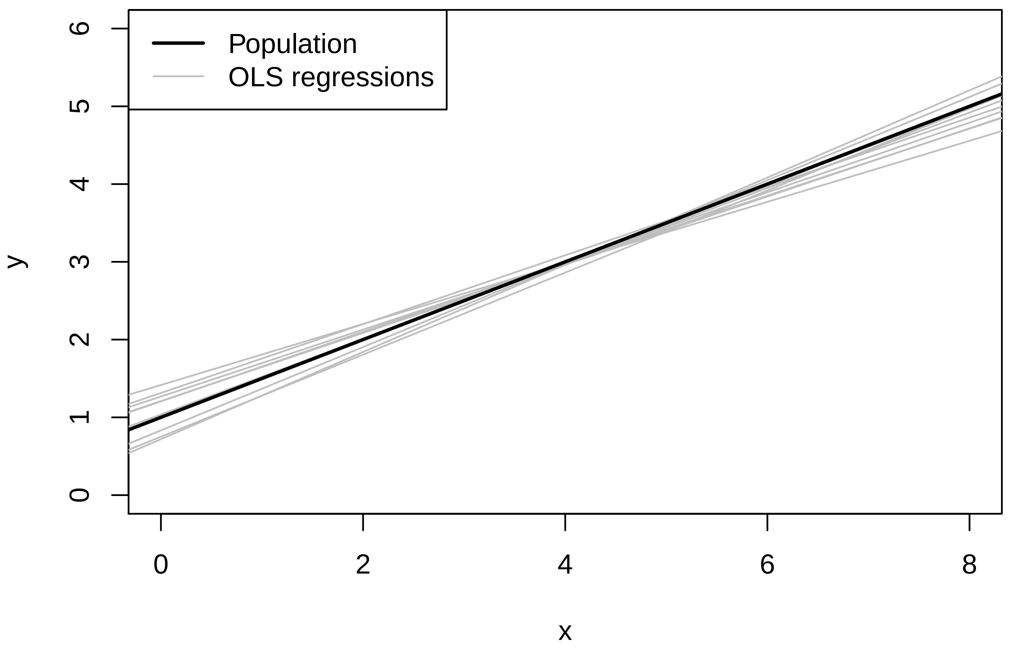


Figure 1.8: Population and Simulated OLS Regression Lines

```
# 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 1.9 shows the simulated distribution of β_0 and β_1 the theoretical one.

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

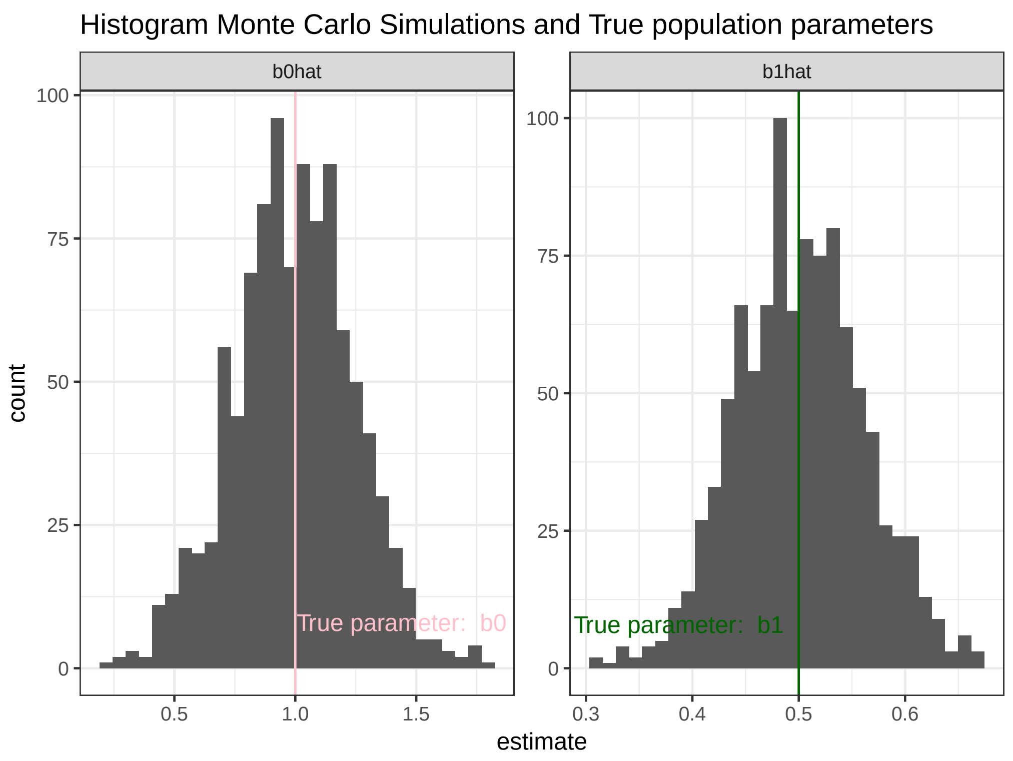


Figure 1.9: Histogram b0 and b1 and true parameter

```
## 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)
```

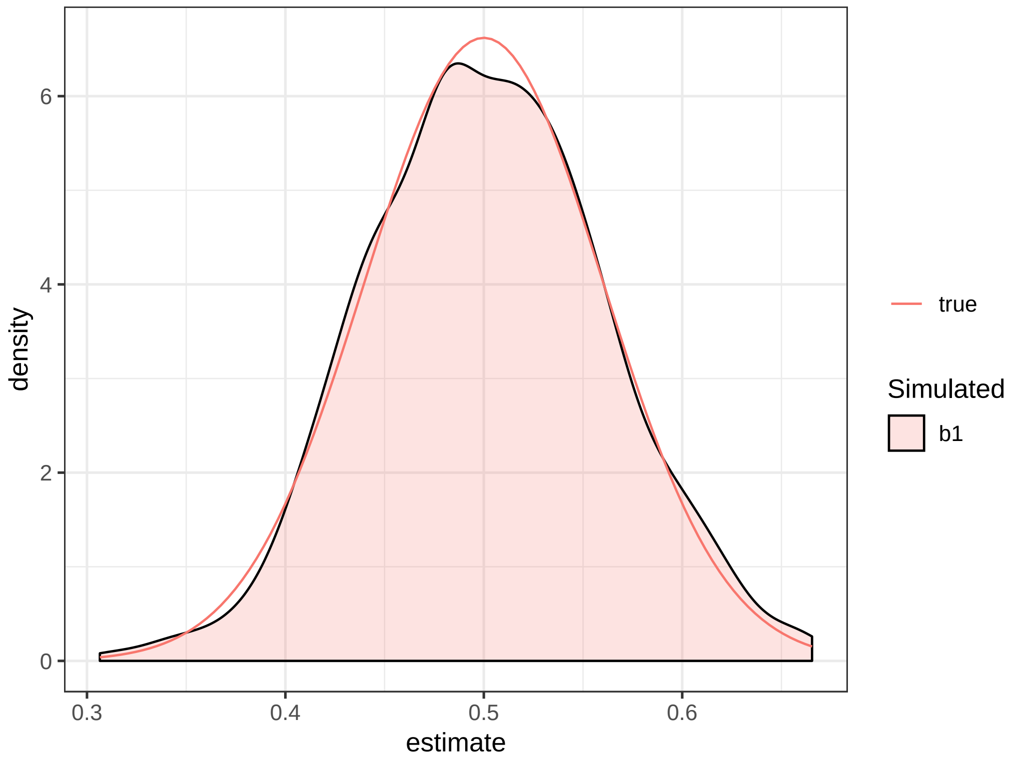


Figure 1.10: Simulated and theoretical distributions of b1

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

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

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

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

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

```r
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="")
```

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

1.1.2.0.2 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

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

```r
# 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:
  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

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

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

```r
mean(b1hat)
```

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

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

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

```r
var(b1hat)
```

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

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

1.1.2.0.3 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.

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

```r
# 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"]
}
```

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

```
##
## 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.

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

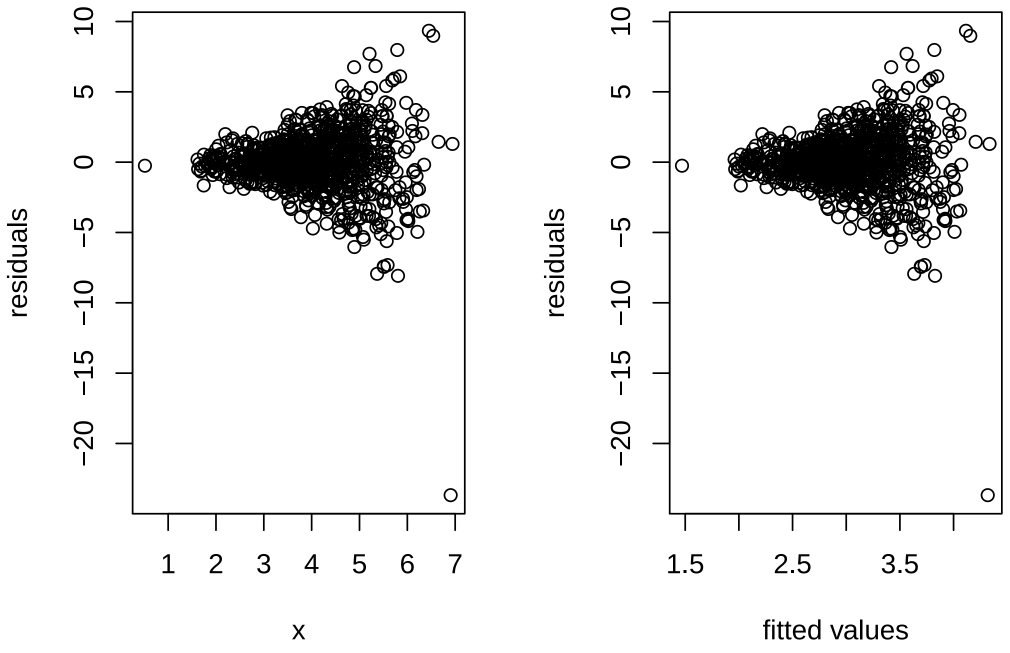


Figure 1.11: Heteroskedasticity in the simulated data

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

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

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

```r
mean(b1hat)
```

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

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

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

```r
var(b1hat)
```

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

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

1.1.3 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.

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

```
## The following objects are masked from wage1 (pos = 17):
##
##     clerocc, construc, educ, exper, expersq, female, lwage,
##     married, ndurman, nonwhite, northcen, numdep, profocc,
##     profserv, services, servocc, smsa, south, tenure, tenursq,
##     trade, trcommpu, wage, west
```

1.1.3.1 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.

```r
# 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)

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 define
```

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

```
## `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:

$$log(Y_i) = \beta_0 + \beta_1 \times x_i + \epsilon_i \tag{1.5}$$

```r
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
```

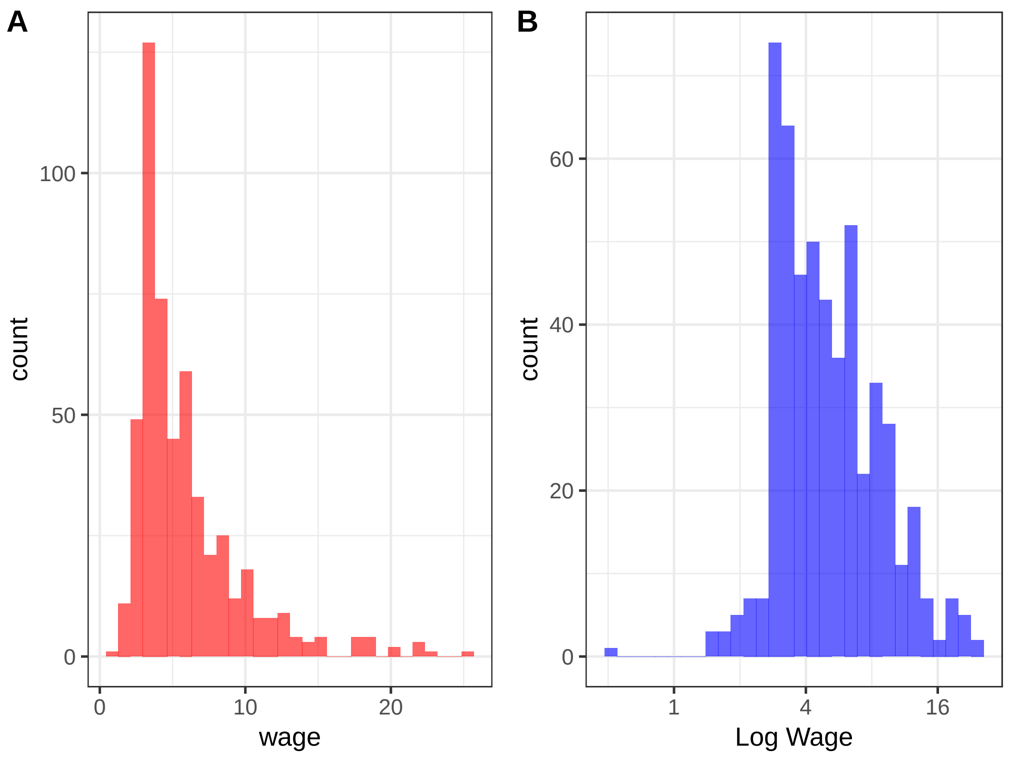


Figure 1.12: Histogram of wage and log(wage)

```
##
## Residual standard error: 3.378 on 524 degrees of freedom
## 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

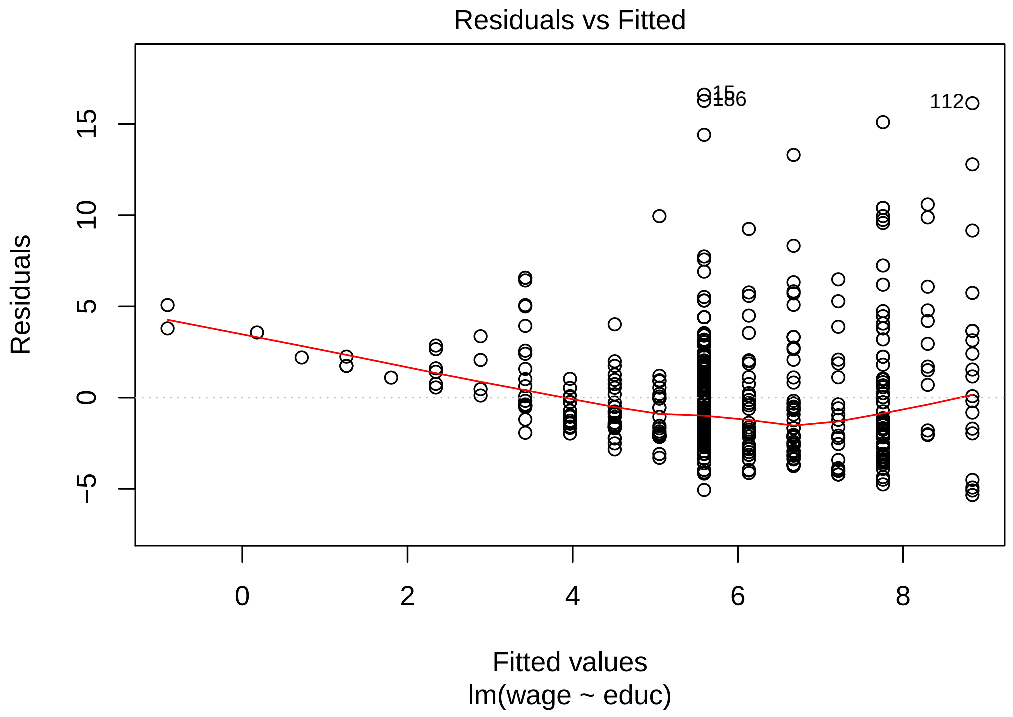


Figure 1.13: Regression diagnostics plot base R - Linear Relationship

```r
plot(lm_wage)
```

```r
autoplot(lm_wage, 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()
```

```r
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()
```

```r
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

$$Y_i = exp(\beta_0 + \beta_1 \times x_i) \times exp(\epsilon_i) \tag{1.6}$$

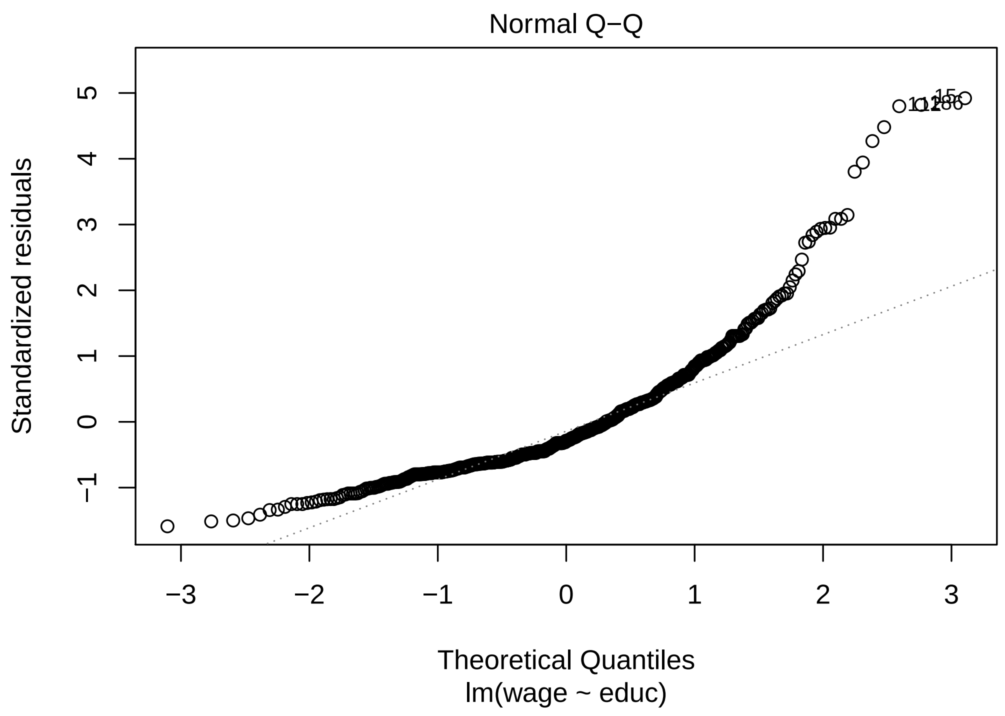


Figure 1.14: Regression diagnostics plot base R - Linear Relationship

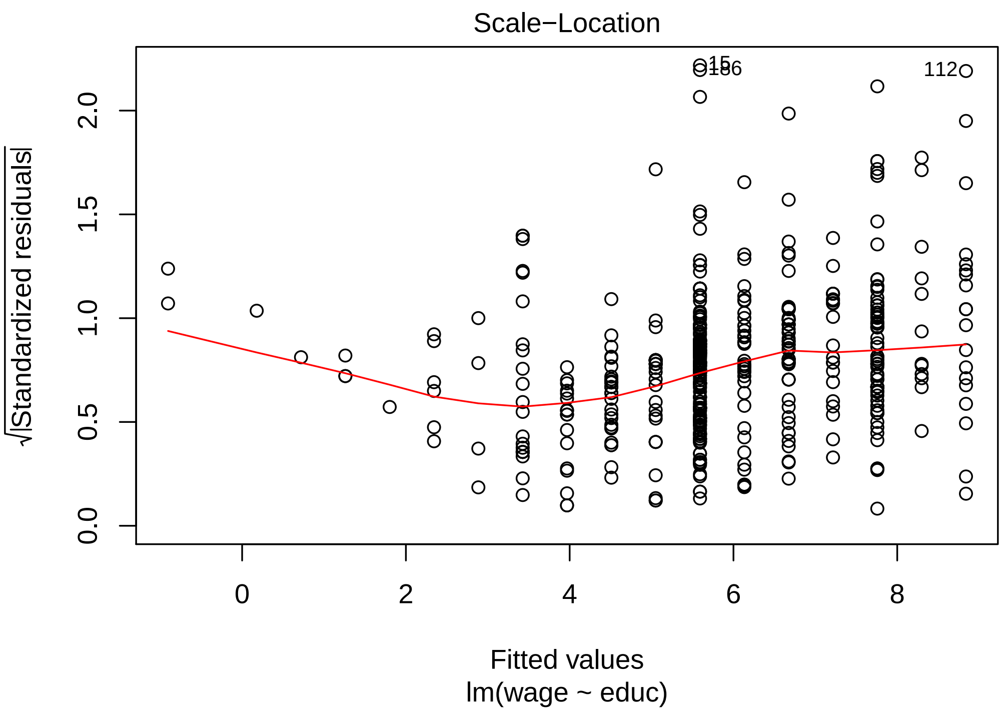


Figure 1.15: Regression diagnostics plot base R - Linear Relationship

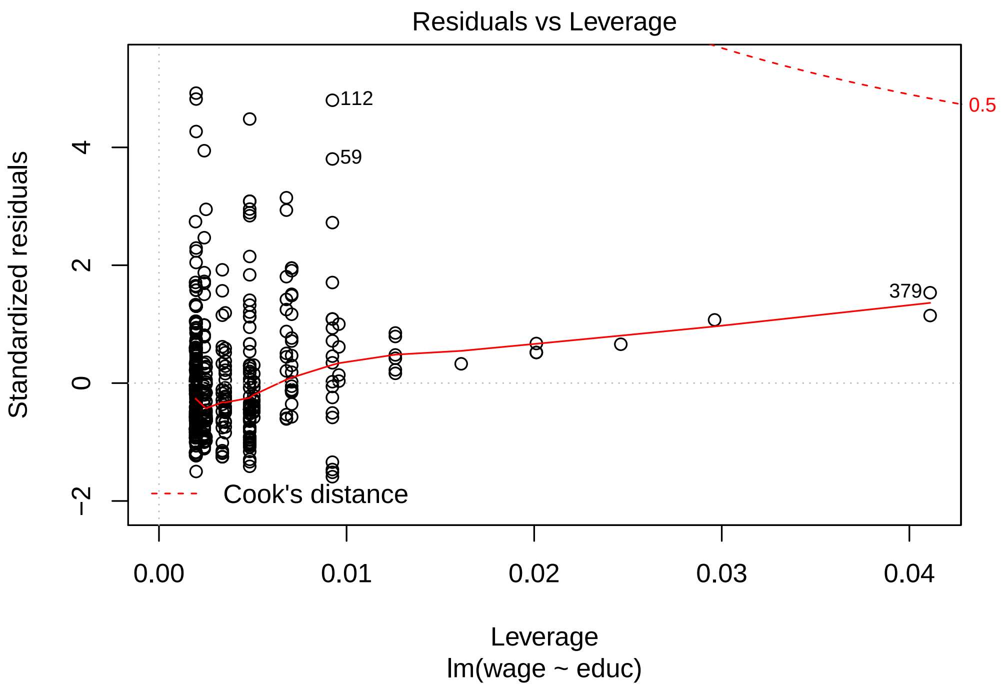


Figure 1.16: Regression diagnostics plot base R - Linear Relationship

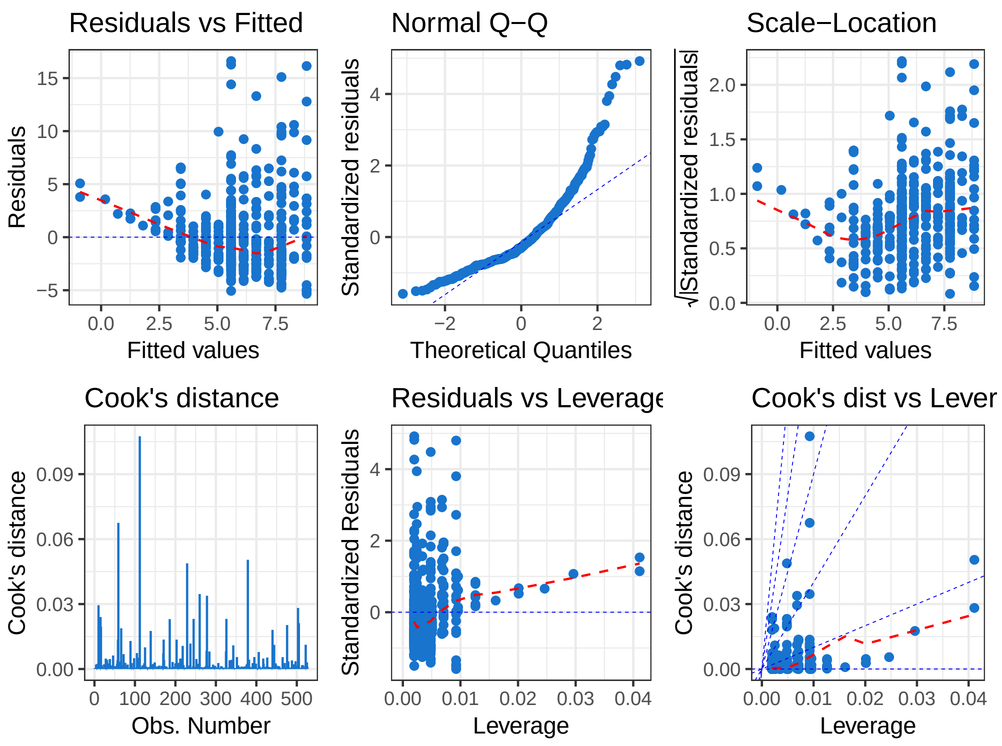


Figure 1.17: Regression diagnostics autoplot(ggplot) - Linear Relationship

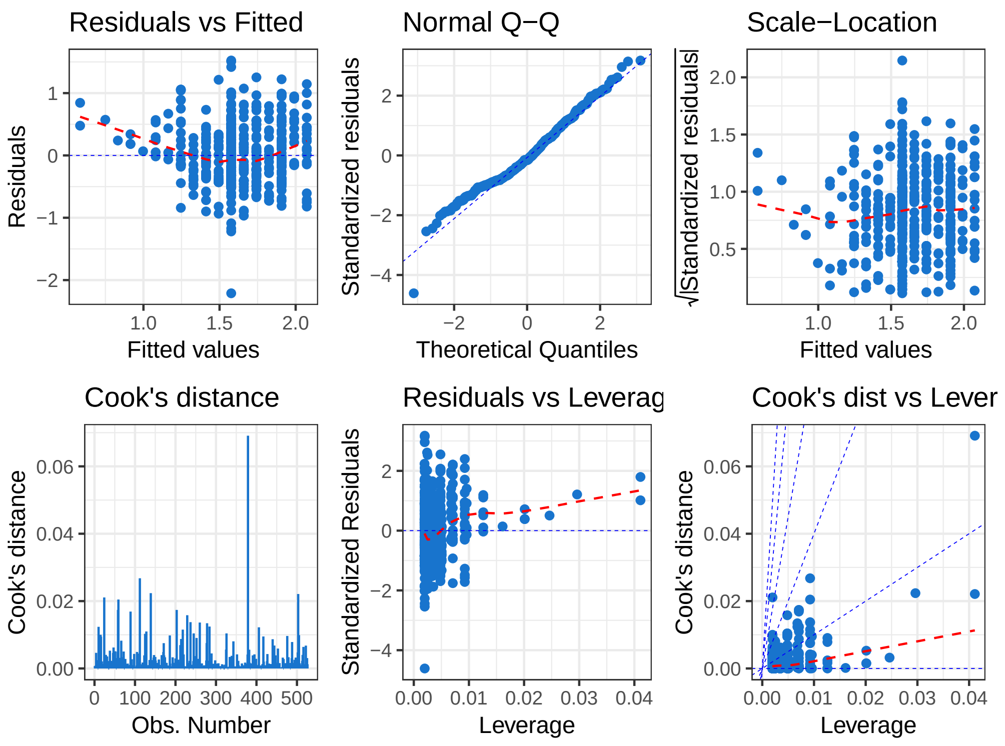


Figure 1.18: Regression diagnostics - Non-Linear Relationship

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
```

```
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

$$Y_i = \beta_0 + \beta_1 \times x_i^2 \times \epsilon_i \tag{1.7}$$

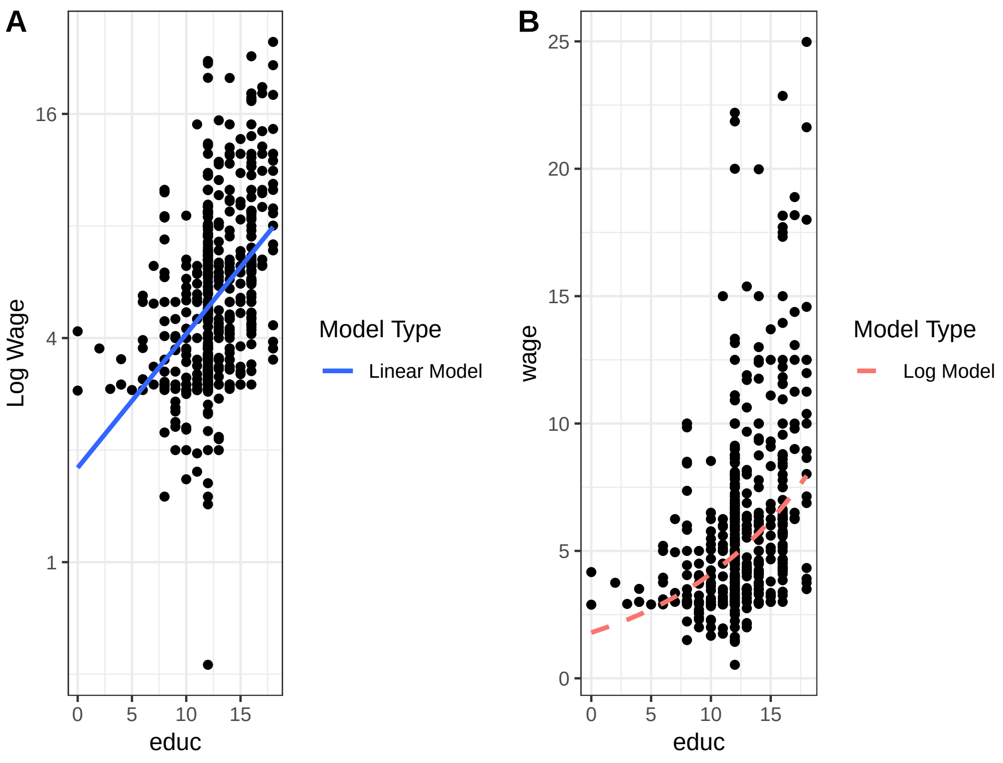


Figure 1.19: Wages by Education - Different transformations

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 = 17):
##
##     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
```

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

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

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

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

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

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

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

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

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

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

1.1.4 Inference for Simple Linear Regression

Note: What should we do in this chapter?

1.2 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 **?**.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + u \tag{1.8}$$

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.

```r
data("gpa1", package = "wooldridge")
attach(gpa1)
```

```
## The following object is masked from bwght:
##
##     male
## The following objects are masked from gpa1 (pos = 17):
##
##     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 object is masked from package:robustbase:
##
##     alcohol

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

Obtain parameter estimates

```r
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",
            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.

1.2.1 Ceteris Paribus Interpretation and Omitted Variable bias

Consider a regression with two explanatory variables

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 \tag{1.9}$$

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

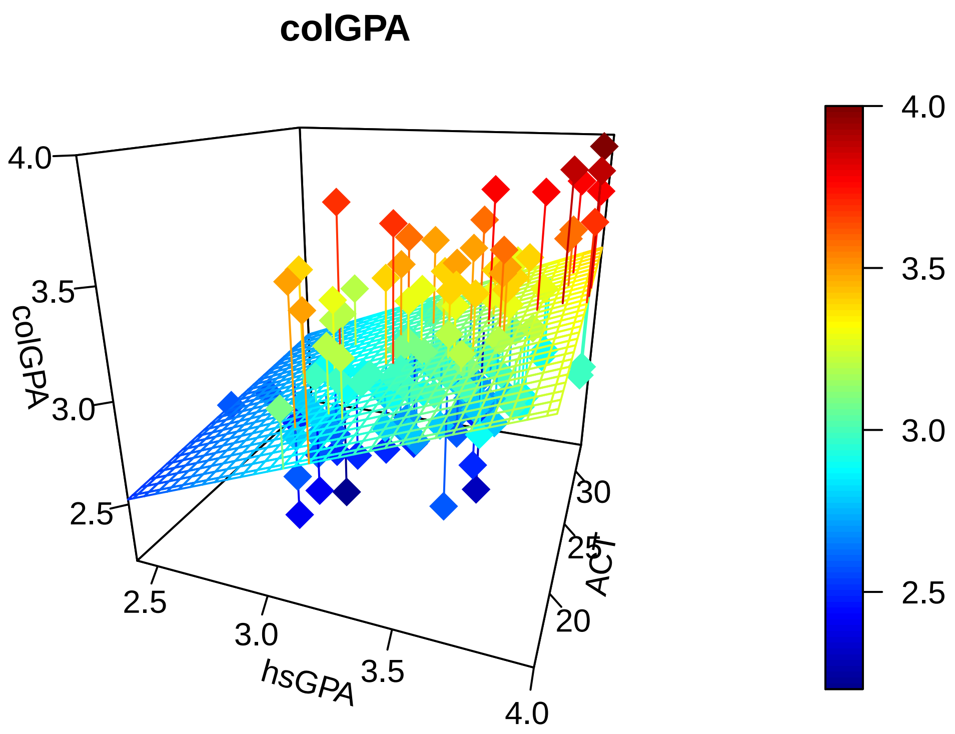


Figure 1.20: College GPA High School GPA + Achievment test score

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

Now, lets ommit one variable in the regression

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 \tag{1.10}$$

```
# 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

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 \tilde{\delta}_1 \tag{1.11}$$

where

$$\hat{x}_2 = \tilde{\delta}_0 + \tilde{\delta}_1 x_1 \tag{1.12}$$

```
# 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*.

1.2.2 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.

$$\frac{1}{1 - R_j^2} \tag{1.13}$$

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

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

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

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

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

```r
vif(GPAres)
```

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

1.2.3 Multiple Regression Analysis: OLS Asymptotics

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

1.2.4 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

Table 1.2: 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

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.

1.2.4.1 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)

```
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

[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.

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

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

Table 1.3

	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		

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

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

1.2.5 Model Formulae

1.2.5.1 Arithmetic operations within a formula

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

```r
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 = 17):
##
##     bwght, bwghtlbs, cigprice, cigs, cigtax, faminc, fatheduc,
##     lbwght, lfaminc, male, motheduc, packs, parity, white

## The following object is masked from gpa1 (pos = 18):
##
##     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:regressiontable) Regression table')  # #tab:foo allows to reference to a table dir
```

```
invisible(stargazer( # invisible supresses additional output such as the package author name when the r
  list(lm1,
       lm2,
       lm3)
  ,keep.stat = c("n", "rsq"), type = "latex", header = FALSE))# to have number of observations and R^2
```

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

1.2.5.2 Standardization: Beta coefficients

The standardized dependent variable y and regressor x_1 are

$$z_y = \frac{y - \bar{y}}{sd(y)} \tag{1.14}$$

and

$$z_{x1} = \frac{x_1 - \bar{x}_{x1}}{sd(x_1)} \tag{1.15}$$

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.

Table 1.4

	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

```
data(hprice2)
```

```
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
```

1.2.5.3 Logarithms, Quadratics and Polynomials

The model for houseprices as in Wooldrige:

$$log(price) = \beta_0 + \beta_1 log(nox) + \beta_2 log(dist) + \beta_3 rooms + \beta_4 rooms^2 + \beta_5 stratio + u \tag{1.16}$$

```
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

$$rooms^{\star} = \frac{-\beta_3}{2\beta_4} \approx 4.4 \tag{1.17}$$

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

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

1.2.5.4 Interaction terms

Consider the following model,

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + u \tag{1.18}$$

where x_1, x_2, and Y are the same as before, but we have added a new interaction term $x_1 x_2$ 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 **x1** times **x2** as $y \sim x_1 + x_2 + x_1 x_2$ and **x1:x2** as $y \sim x_1 x_2$.

```
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:
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

1.2.6 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
```

```
# 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
```

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

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

```
# 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

```
# 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
```

1.2.6.1 Prediction intervals

```
# 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

1.3 MLR Analysis with Qualitative Regressors

1.3.1 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
```

1.3.2 Logical variables

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

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

```
# 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
```

1.3.3 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.

```
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 ...
```

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

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

```
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
```

1.3.3.1 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)
```

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		

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%.

1.3.4 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
```

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

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

1.3.4.1 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")
```

1.4 Heteroskedasticity

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

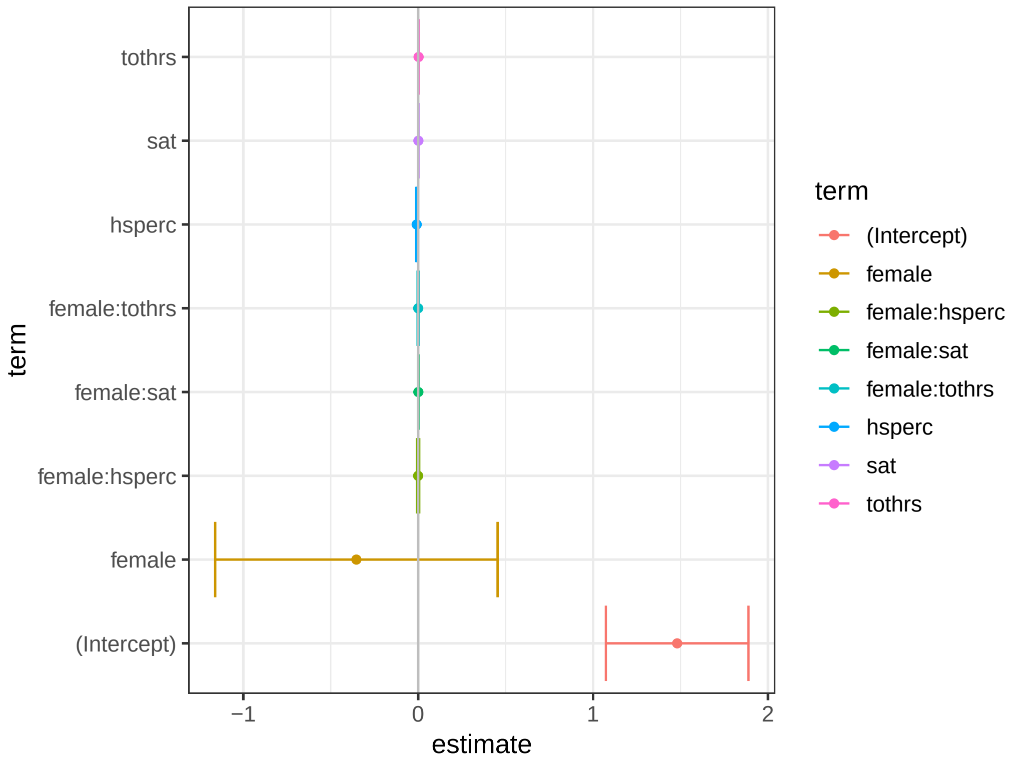


Figure 1.21: Coefficient plots

$$Var(u|x_1, \ldots, x_n) = \sigma^2 \tag{1.19}$$

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 **?**.

1.4.1 Spotting Heteroskedasticity in Scatter Plots

```r
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.

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

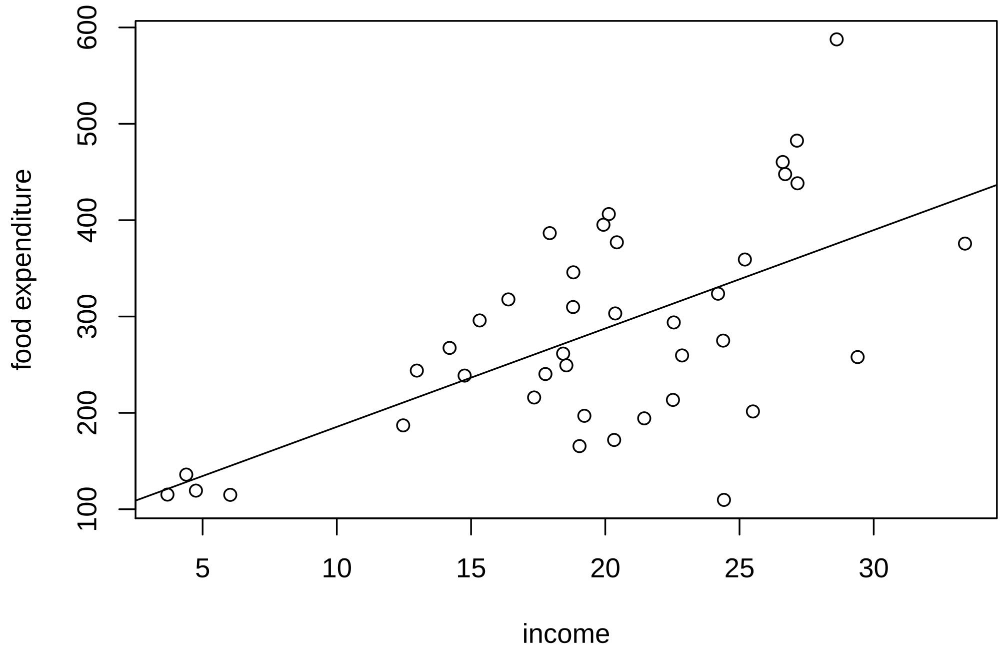


Figure 1.22: Heteroskedasticity in the ‘food’ data

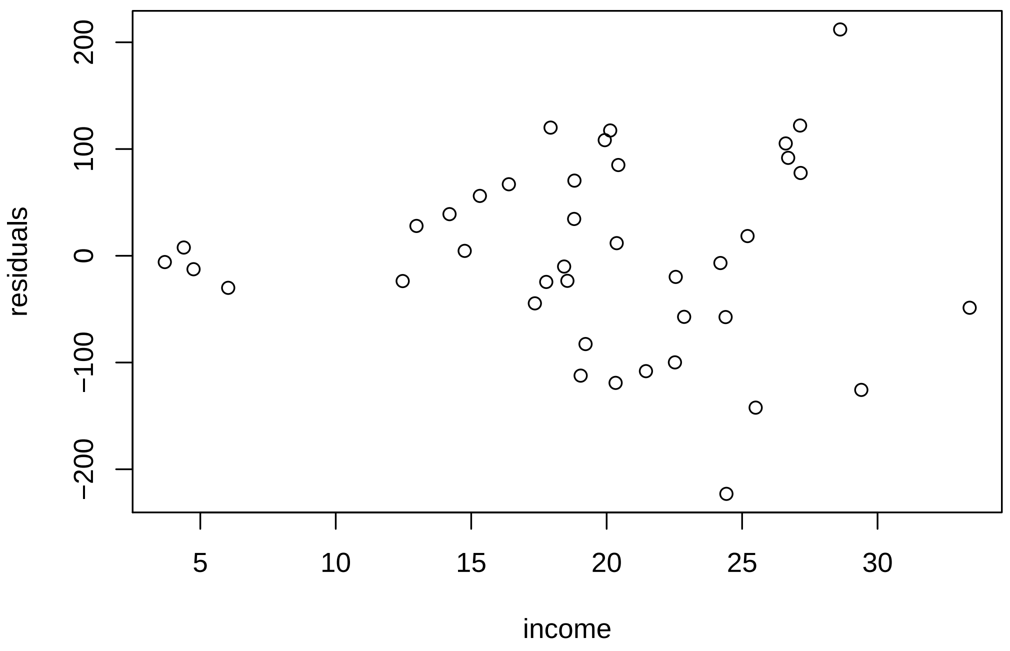


Figure 1.23: Residual plots in the ‘food’ model

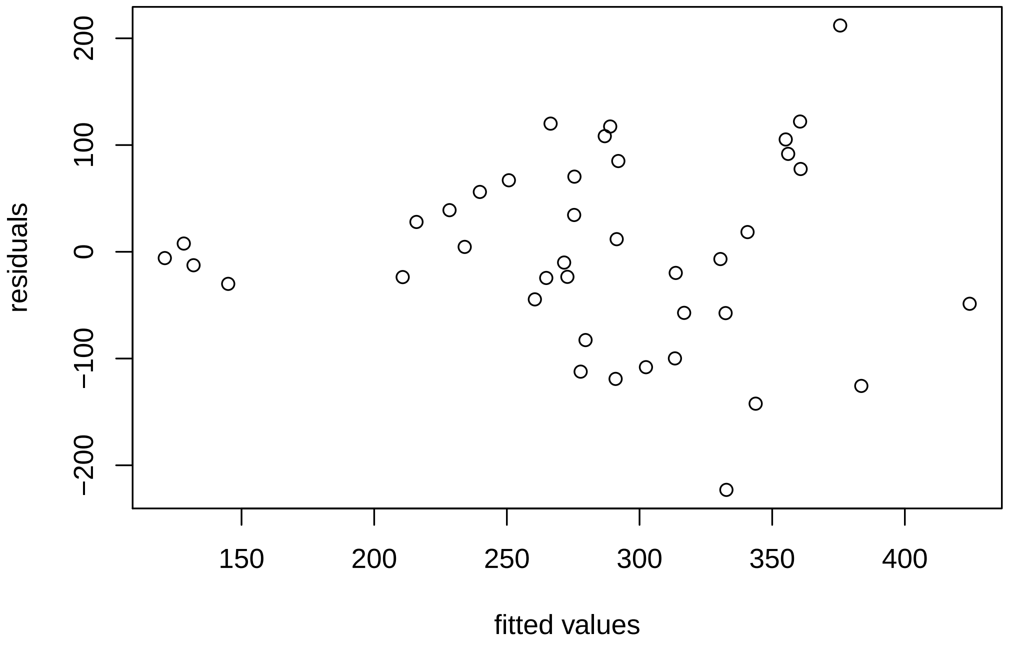


Figure 1.24: Residual plots in the 'food' model

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

1.4.2 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()`:

```
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"
```

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

```
## 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
##         1.000e-07         1.000e-10         1.000e-07         2.732e-04
##             eps.x warn.limit.reject warn.limit.meanrw
##         2.601e-09         5.000e-01         5.000e-01
##      nResample         max.it       best.r.s       k.fast.s          k.max
##            500             50              2              1            200
##    maxit.scale      trace.lev            mts     compute.rd fast.s.large.n
##            200              0           1000              0           2000
##                   psi           subsampling                   cov
##             "bisquare"         "nonsingular"         ".vcov.avar1"
## compute.outlier.stats
##                  "SM"
## seed : int(0)
```

This however, gives you different estimates than when fitting a linear regression model. The estimates should be the same, only the standard errors should be different. This is because the estimation method is different, and is also robust to outliers (at least that's my understanding, I haven't read the theoretical papers behind the package yet).

Finally, it is also possible to bootstrap the standard errors. For this I will use the `bootstrap()` function from the modelr package:

```
resamples <- 100

boot_gpa3 <- gpa3 %>%
  modelr::bootstrap(resamples)
```

The column strap contains resamples of the original data. I will run my linear regression from before on each of the resamples:

```
boot_lin_reg <- boot_gpa3 %>%
  mutate(regressions =
           map(strap,
               ~lm(cumgpa~sat+hsperc+tothrs+female+black+white,
                   data= . , subset=(spring==1)))
  )
```

We have added a new column called regressions which contains the linear regressions on each bootstrapped sample. Now, I will create a list of tidied regression results:

```
tidied <- boot_lin_reg %>%
  mutate(tidy_lm =
           map(regressions, broom::tidy))
```

```
tidied$tidy_lm[[1]]
```

term	estimate	std.error	statistic	p.value
(Intercept)	1.43	0.247	5.8	1.43e-08
sat	0.000946	0.000193	4.91	1.39e-06
hsperc	-0.00908	0.00131	-6.91	2.08e-11
tothrs	0.00452	0.000737	6.13	2.23e-09
female	0.222	0.062	3.59	0.000379
black	-0.0733	0.164	-0.447	0.655
white	0.063	0.16	0.395	0.693

```
list_mods <- tidied %>%
  pull(tidy_lm)
```

```
mods_df <- map2_df(list_mods,
                   seq(1, resamples),
                   ~mutate(.x, resample = .y))
```

```
head(mods_df, 5)
```

term	estimate	std.error	statistic	p.value	resample
(Intercept)	1.43	0.247	5.8	1.43e-08	1
sat	0.000946	0.000193	4.91	1.39e-06	1
hsperc	-0.00908	0.00131	-6.91	2.08e-11	1
tothrs	0.00452	0.000737	6.13	2.23e-09	1
female	0.222	0.062	3.59	0.000379	1

```
r.std.error <- mods_df %>%
  group_by(term) %>%
  summarise(r.std.error = sd(estimate))
```

```
reg %>%
  broom::tidy()   %>%
  full_join(r.std.error)
```

```
## Joining, by = "term"
```

term	estimate	std.error	statistic	p.value	r.std.error
(Intercept)	1.47	0.23	6.4	4.94e-10	0.218
sat	0.00114	0.000179	6.39	5.2e-10	0.000168
hsperc	-0.00857	0.00124	-6.91	2.27e-11	0.00137
tothrs	0.0025	0.000731	3.43	0.000685	0.000782
female	0.303	0.059	5.14	4.5e-07	0.0575
black	-0.128	0.147	-0.87	0.385	0.135
white	-0.0587	0.141	-0.416	0.677	0.111

Using the whole bootstrapping procedure is longer than simply using either one of the first two methods. However, this procedure is very flexible and can thus be adapted to a very large range of situations.

1.5 Weighted least squares

Weighted Least Squares (WLS) attempts to provide a more efficient alternative to OLS. It is a special version of a feasbile generalized least squares (FGLS) estimator.

```
data("k401k")
```

```
# OLS (only for singles: fsize==1)
lm(nettfa ~ inc + I((age-25)^2) + male + e401k,
   data=k401ksubs, subset=(fsize==1))
```

```
##
## Call:
## lm(formula = nettfa ~ inc + I((age - 25)^2) + male + e401k, data = k401ksubs,
##     subset = (fsize == 1))
##
## Coefficients:
##     (Intercept)              inc  I((age - 25)^2)             male
##       -20.98499          0.77058          0.02513          2.47793
##           e401k
##         6.88622
```

Following Wooldrige, we assume that the variance is proportional to the income variable *inc*.. Therefore, the optimal weight is $\frac{1}{inc}$ which is given as *weight* in the `lm()` call.

```
# WLS
lm(nettfa ~ inc + I((age-25)^2) + male + e401k, weight=1/inc,
   data=k401ksubs, subset=(fsize==1))
```

```
##
## Call:
## lm(formula = nettfa ~ inc + I((age - 25)^2) + male + e401k, data = k401ksubs,
##     subset = (fsize == 1), weights = 1/inc)
##
## Coefficients:
```

```
##     (Intercept)              inc  I((age - 25)^2)             male
##       -16.70252          0.74038          0.01754          1.84053
##           e401k
##         5.18828
```

We can also use heteroscedasticity-robust statistics to account for the fact that our variance function might be misspecified.

```
# WLS
wlsreg <- lm(nettfa ~ inc + I((age-25)^2) + male + e401k,
             weight=1/inc, data=k401ksubs, subset=(fsize==1))

# non-robust results
coeftest(wlsreg)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)     -16.7025205   1.9579947 -8.5304 < 2.2e-16 ***
## inc               0.7403843   0.0643029 11.5140 < 2.2e-16 ***
## I((age - 25)^2)   0.0175373   0.0019315  9.0796 < 2.2e-16 ***
## male              1.8405293   1.5635872  1.1771  0.239287
## e401k             5.1882807   1.7034258  3.0458  0.002351 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# robust results (Refined White SE:)
coeftest(wlsreg,hccm)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)     -16.7025205   2.2482355 -7.4292 1.606e-13 ***
## inc               0.7403843   0.0752396  9.8403 < 2.2e-16 ***
## I((age - 25)^2)   0.0175373   0.0025924  6.7650 1.742e-11 ***
## male              1.8405293   1.3132477  1.4015 0.1612159
## e401k             5.1882807   1.5743329  3.2955 0.0009994 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(wlsreg, vcov. = vcovHC)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)     -16.7025205   2.2482355 -7.4292 1.606e-13 ***
## inc               0.7403843   0.0752396  9.8403 < 2.2e-16 ***
## I((age - 25)^2)   0.0175373   0.0025924  6.7650 1.742e-11 ***
## male              1.8405293   1.3132477  1.4015 0.1612159
## e401k             5.1882807   1.5743329  3.2955 0.0009994 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mySummary <- function(model, VCOV) {
  print(coeftest(model, vcov. = VCOV))
  print(waldtest(model, vcov = VCOV))
}
mySummary(wlsreg, VCOV = vcovHAC)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)     -16.7025205   2.2425229 -7.4481 1.397e-13 ***
## inc               0.7403843   0.0752621  9.8374 < 2.2e-16 ***
## I((age - 25)^2)   0.0175373   0.0025797  6.7981 1.392e-11 ***
## male              1.8405293   1.3056244  1.4097  0.158785
## e401k             5.1882807   1.5733280  3.2976  0.000992 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Wald test
##
## Model 1: nettfa ~ inc + I((age - 25)^2) + male + e401k
## Model 2: nettfa ~ 1
##   Res.Df Df      F    Pr(>F)
## 1   2012
## 2   2016 -4 39.602 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The asumption that the variance is proportional to a regressor is usually hard to justify. Typically, we do not know the variance function; we have to estimate it. We can estiamte the relation between variance and regressors using a linear regression of the log of the squared residuals from an initial OlS regression $log(\hat{u}^2)$ as the dependent variable.

Wooldrige suggests two version for the selection of regressors:

- the regressors $x_1, \ldots, x_k$ from the original model similar to the BP test
- $\hat{y}$ and $\hat{y}^2$ from the original model similar to the White test

```r
data("smoke")
# OLS
olsreg<-lm(cigs~log(income)+log(cigpric)+educ+age+I(age^2)+restaurn,
           data=smoke)
olsreg
```

```
##
## Call:
## lm(formula = cigs ~ log(income) + log(cigpric) + educ + age +
##     I(age^2) + restaurn, data = smoke)
##
## Coefficients:
##  (Intercept)   log(income)  log(cigpric)          educ           age
##    -3.639826      0.880268     -0.750862     -0.501498      0.770694
##     I(age^2)      restaurn
##    -0.009023     -2.825085
```

```
# BP test
bptest(olsreg)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  olsreg
## BP = 32.258, df = 6, p-value = 1.456e-05
```

```
# FGLS: estimation of the variance function
logu2 <- log(resid(olsreg)^2)
varreg<-lm(logu2~log(income)+log(cigpric)+educ+age+I(age^2)+restaurn,
           data=smoke)

# FGLS: WLS
w <- 1/exp(fitted(varreg))
lm(cigs~log(income)+log(cigpric)+educ+age+I(age^2)+restaurn,
   weight=w ,data=smoke)
```

```
##
## Call:
## lm(formula = cigs ~ log(income) + log(cigpric) + educ + age +
##     I(age^2) + restaurn, data = smoke, weights = w)
##
## Coefficients:
##  (Intercept)   log(income)  log(cigpric)          educ           age
##     5.635463      1.295239     -2.940312     -0.463446      0.481948
##     I(age^2)      restaurn
##    -0.005627     -3.461064
```

1.6 Model specification and Parameter Heterogeneity

1.6.1 Functional Form Misspecifcation

We have seen many ways to specifiy the relation between the dependent variable and the regressors. An obvious question is to ask whether or not a given specification is "correct".

1.6.1.1 RESET

The Regresion Equation Specification Error Test (RESET) is a convient tool to test the null hypothesis that the functional form is adequate.

We can run the test ourselves or use the boxed routine `resettest()` from the package **lmtest**.

```
data("hprice1")
# original linear regression
orig <- lm(price ~ lotsize+sqrft+bdrms, data=hprice1)

# regression for RESET test
RESETreg <- lm(price ~ lotsize+sqrft+bdrms+I(fitted(orig)^2)+
                 I(fitted(orig)^3), data=hprice1)
RESETreg
```

```
##
## Call:
## lm(formula = price ~ lotsize + sqrft + bdrms + I(fitted(orig)^2) +
##     I(fitted(orig)^3), data = hprice1)
##
## Coefficients:
##       (Intercept)            lotsize              sqrft
##         1.661e+02          1.537e-04          1.760e-02
##             bdrms  I(fitted(orig)^2)  I(fitted(orig)^3)
##         2.175e+00          3.534e-04          1.546e-06
```

```
# RESET test. H0: all coeffs including "fitted" are=0
linearHypothesis(RESETreg, matchCoefs(RESETreg,"fitted"))
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
84	3.01e+05				
82	2.7e+05	2	3.07e+04	4.67	0.012

Automatic routine:

```
# original linear regression
orig <- lm(price ~ lotsize+sqrft+bdrms, data=hprice1)

# RESET test
resettest(orig)
```

```
##
##  RESET test
##
## data:  orig
## RESET = 4.6682, df1 = 2, df2 = 82, p-value = 0.01202
```

Wooldrige (2016, Section 9.2) also disucsses tests of non-nested models. We can use the `encomptest()` function from the package **lmtest**.

Two alternative models for the housing price

$$price = \beta_0 + \beta_1 lotsize + \beta_2 sqrft + \beta_3 bdrms + u \tag{1.20}$$

$$price = \beta_0 + \beta_1 log(lotsize) + \beta_2 log(sqrft) + \beta_3 log(bdrms) + u \tag{1.21}$$

```
# two alternative models
model1 <- lm(price ~     lotsize  +     sqrft  + bdrms, data=hprice1)
model2 <- lm(price ~ log(lotsize) + log(sqrft) + bdrms, data=hprice1)

# Test against comprehensive model
encomptest(model1,model2, data=hprice1)
```

Res.Df	Df	F	Pr(>F)
82	-2	7.86	0.000753
82	-2	7.05	0.00149

The ouput shows the "encompassing model" E with all variables. Both models are rejected against this comprehensive model.

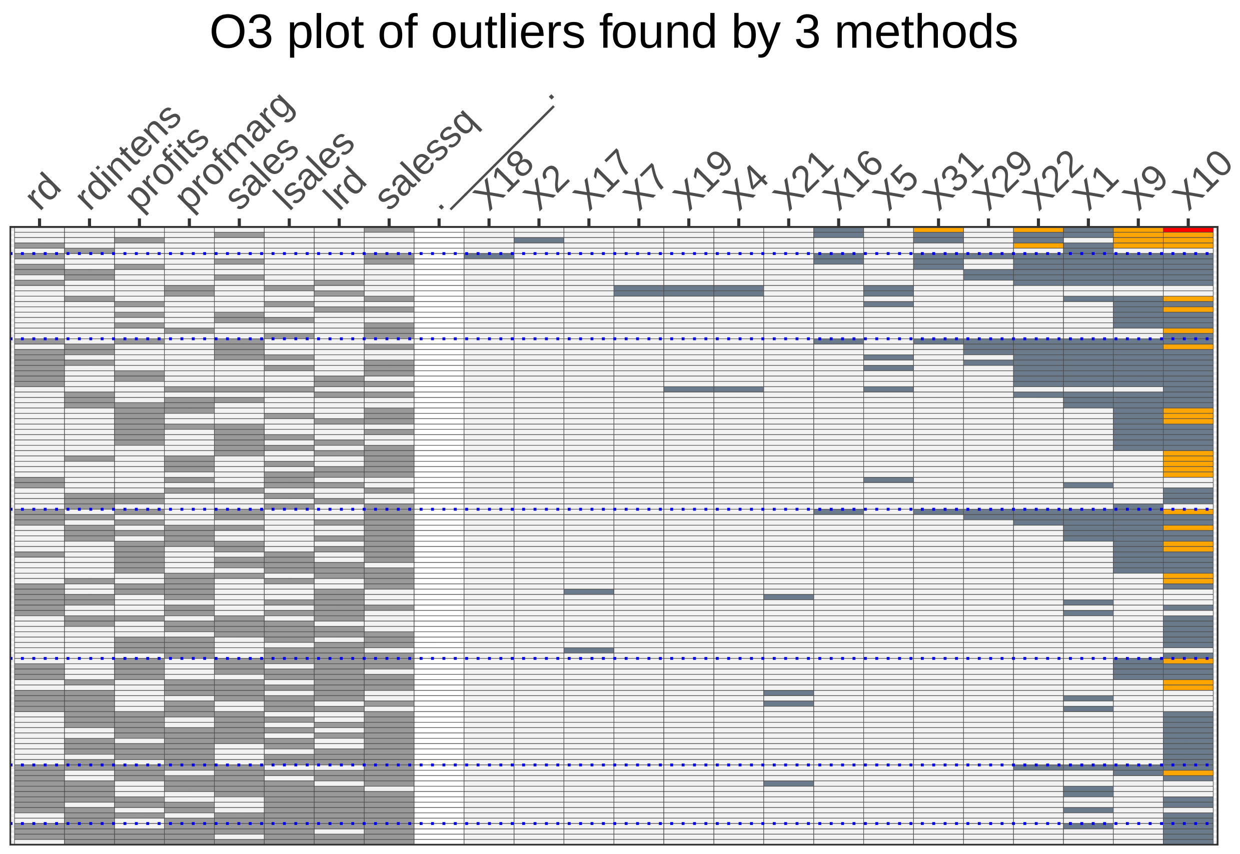
1.6.1.2 Outlying observations

Dealing with outliers is a tricky business. R offers different pacakges to test and ajust for outliers. But outliers can be a matter of opinion and not all outlier dedection methods give the same results. With the **OutliersO3** package we can compare different outlier dedection methods.

```
data(rdchem)

s3 <- O3prep(rdchem, method=c("HDo", "adjOut", "DDC"))
O3s3 <- O3plotM(s3)
print(O3s3$nOut)
```

```
##    HDo adjOut    DDC
##     10      1     11
```

```
O3s3$gO3
```



An O3 plot of stackloss using the methods HDoutliers, adjOutlyingness and DectectDeviatingCells. The darker the cell, the more methods agree. If they all agree, the cell is coloured red and if all but one agree then orange.

We use functions from the **car** package to obtain a table of different measures of leverage and influence for all observations.

```
# Regression
reg <- lm(rdintens~sales+profmarg, data=rdchem)
```

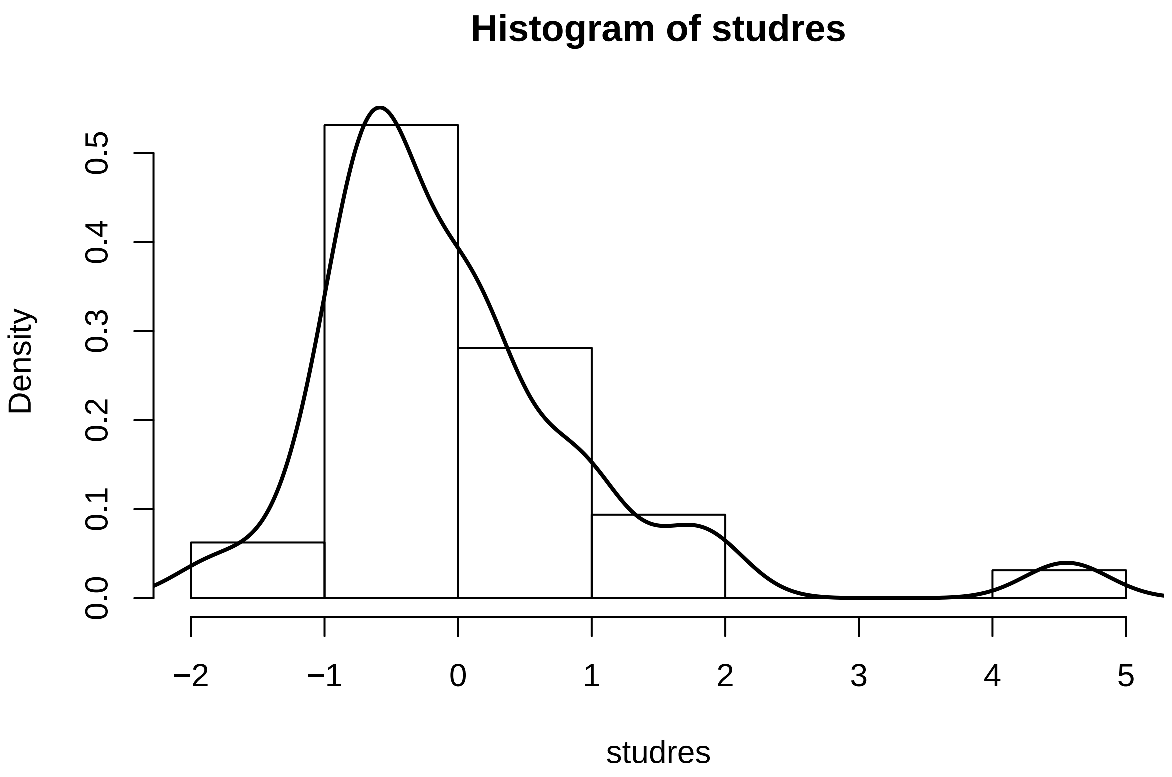
```
# Studentized residuals for all observations:
studres <- rstudent(reg)

# Display extreme values:
min(studres)
```

```
## [1] -1.818039
```

```
max(studres)
```

```
## [1] 4.555033
```

```
# Histogram (and overlayed density plot):
hist(studres, freq=FALSE)
lines(density(studres), lwd=2)
```



1.6.1.3 Missing Data

Missing values in data is a common phenomenon in real world problems. In R, missing data can be represented by different values of the variable.

- **NA** (not available) indicates that we do not have the information
- **NaN** (not a number) indicates that the value is not defined, for example when we take the log of a negative number

Base R offers many functions to dedect missing observations. Sometimes using **mice** and **VIM** package for looking at missing data pattern and imputing missing data is even easier.

```
data("lawsch85", package = "wooldridge" )

# extract LSAT
```

```
lsat <- lawsch85$LSAT

# Create logical indicator for missings
missLSAT <- is.na(lawsch85$LSAT)

# LSAT and indicator for Schools No. 120-129:
rbind(lsat,missLSAT)[,120:129]
```

```
##          [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## lsat      156  159  157  167   NA  158  155  157   NA   163
## missLSAT    0    0    0    0    1    0    0    0    1     0
```

```
# Frequencies of indicator
table(missLSAT)
```

```
## missLSAT
## FALSE  TRUE
##   150     6
```

```
# Missings for all variables in data frame (counts)
colSums(is.na(lawsch85))
```

```
##    rank  salary    cost    LSAT     GPA  libvol faculty     age  clsize
##       0       8       6       6       7       1       4      45       3
##   north   south    east    west lsalary studfac   top10  r11_25  r26_40
##       0       0       0       0       8       6       0       0       0
##  r41_60 llibvol   lcost
##       0       1       6
```

```
# Indicator for complete cases
compl <- complete.cases(lawsch85)
table(compl)
```

```
## compl
## FALSE  TRUE
##    66    90
```

```
# MICE package function to display msising values
head(md.pattern(lawsch85, plot = FALSE))
```

```
##    rank north south east west top10 r11_25 r26_40 r41_60 libvol llibvol
## 90    1     1     1    1    1     1      1      1      1      1       1
## 41    1     1     1    1    1     1      1      1      1      1       1
## 6     1     1     1    1    1     1      1      1      1      1       1
## 1     1     1     1    1    1     1      1      1      1      1       1
## 3     1     1     1    1    1     1      1      1      1      1       1
## 1     1     1     1    1    1     1      1      1      1      1       1
##    clsize faculty cost LSAT studfac lcost GPA salary lsalary age
## 90      1       1    1    1       1     1   1      1       1   1 0
## 41      1       1    1    1       1     1   1      1       1   0 1
## 6       1       1    1    1       1     1   1      0       0   1 2
## 1       1       1    1    1       1     1   1      0       0   0 3
## 3       1       1    1    0       1     1   0      1       1   1 2
## 1       1       1    1    0       1     1   0      1       1   0 3
```

```
aggr_plot <- aggr(lawsch85, col=c('navyblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(lawsch85
```

```
##
```

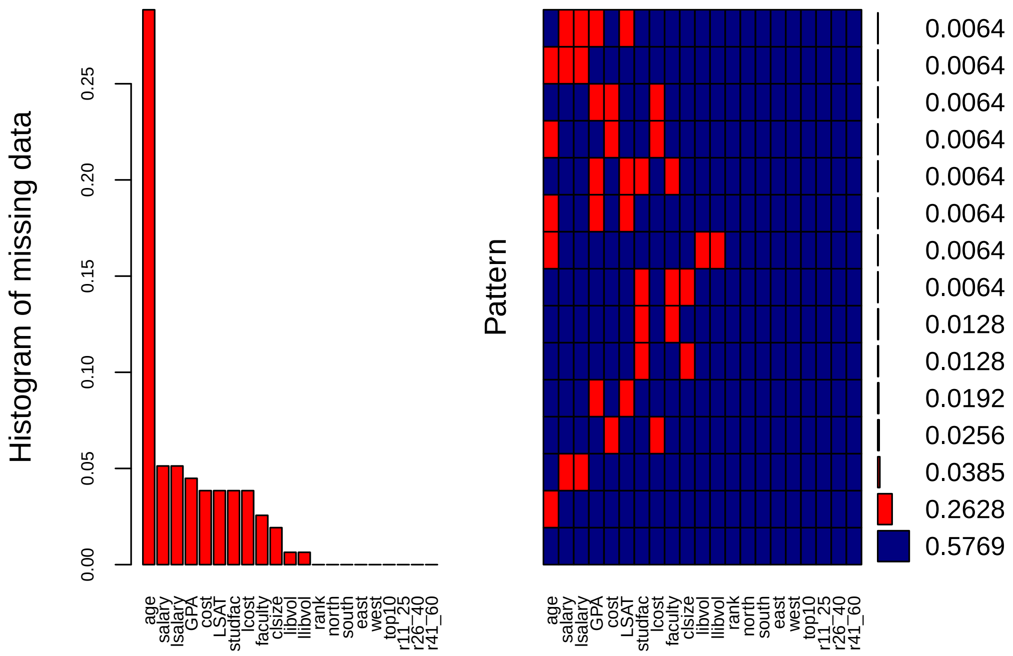


Figure 1.25: Visualizing missing data

```
##  Variables sorted by number of missings:
##  Variable       Count
##       age 0.288461538
##    salary 0.051282051
##   lsalary 0.051282051
##       GPA 0.044871795
##      cost 0.038461538
##      LSAT 0.038461538
##   studfac 0.038461538
##     lcost 0.038461538
##   faculty 0.025641026
##    clsize 0.019230769
##    libvol 0.006410256
##   llibvol 0.006410256
##      rank 0.000000000
##     north 0.000000000
##     south 0.000000000
##      east 0.000000000
##      west 0.000000000
##     top10 0.000000000
##    r11_25 0.000000000
##    r26_40 0.000000000
##    r41_60 0.000000000
```

Regression command like `lm()` have as argument **na.rm=TRUE**!

```
# Mean of a variable with missings:
mean(lawsch85$LSAT)
```

```
## [1] NA
```

```
mean(lawsch85$LSAT,na.rm=TRUE)
```

```
## [1] 158.2933
```

```
# Regression with missings
summary(lm(log(salary)~LSAT+cost+age, data=lawsch85))
```

```
##
## Call:
## lm(formula = log(salary) ~ LSAT + cost + age, data = lawsch85)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40989 -0.09438  0.00317  0.10436  0.45483
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.384e+00  6.781e-01   6.465 4.94e-09 ***
## LSAT        3.722e-02  4.501e-03   8.269 1.06e-12 ***
## cost        1.114e-05  4.321e-06   2.577 0.011563 *
## age         1.503e-03  4.354e-04   3.453 0.000843 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1545 on 91 degrees of freedom
##   (61 observations deleted due to missingness)
## Multiple R-squared:  0.6708, Adjusted R-squared:  0.6599
## F-statistic: 61.81 on 3 and 91 DF,  p-value: < 2.2e-16
```

R packages provide multiple imputation algorithms. Without going into detail how those algorithms work, we can use for example the **meth="pmm"** argument from the **mice** package to apply a predictive mean matching as imputation method.

```
# We use a diffferent dataset to speed up the imputation process
data <- airquality
data[4:10,3] <- rep(NA,7)
data[1:5,4] <- NA
tempData <- mice(data,m=5,maxit=50,meth='pmm',seed=500)
```

```
summary(tempData)
```

```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##   Ozone Solar.R    Wind    Temp   Month     Day
##   "pmm"   "pmm"   "pmm"   "pmm"      ""      ""
## PredictorMatrix:
##         Ozone Solar.R Wind Temp Month Day
## Ozone       0       1    1    1     1   1
## Solar.R     1       0    1    1     1   1
## Wind        1       1    0    1     1   1
## Temp        1       1    1    0     1   1
## Month       1       1    1    1     0   1
## Day         1       1    1    1     1   0
```

1.7 Least absolute Deviations (LAD) Estimation

As an alterantive to OLS, the least absolute deviations (LAD) is less sensitive to outliers. Instead of minimizing the sum of *squared* residuals, it minimizes the sum of the *absolute values* of the residuals.

In R, general quantile regression (and LAD as the default special case) can easily be implemented with the command `reg()` from the **quantreg** package.

```
# OLS Regression
ols <- lm(rdintens ~ I(sales/1000) +profmarg, data=rdchem)
# LAD Regression
lad <- rq(rdintens ~ I(sales/1000) +profmarg, data=rdchem)

# regression table
stargazer(ols,lad,  type = "text")
```

```
##
## ==================================================
##                          Dependent variable:
##                     ------------------------------
##                                rdintens
##                           OLS          quantile
##                                       regression
##                           (1)             (2)
## --------------------------------------------------
## I(sales/1000)            0.053           0.019
##                         (0.044)         (0.059)
##
## profmarg                 0.045          0.118**
##                         (0.046)         (0.049)
##
## Constant                2.625***        1.623***
##                         (0.586)         (0.509)
##
## --------------------------------------------------
## Observations               32              32
## R2                       0.076
## Adjusted R2              0.012
## Residual Std. Error  1.862 (df = 29)
## F Statistic         1.195 (df = 2; 29)
## ==================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

Note: LAD inferences are only valid asymptotically, so the results in this example with $n = 32$ should be taken with caution.

Chapter 2

Qualitative and LDV Models

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
            "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,
            "MCMCpack", # Contains functions to perform Bayesian inference using posterior simulation f
            "sampleSelection", # Two-step and maximum likelihood estimation of Heckman-type sample sele
            "scales", # scale helper functions such as percent
            "lmtest",
            "margins", # Stata like margin functions
            "prediction", # Type stable predictions
            "nnet", # Multinomial logit
            "survival", # Survival Analysis
            "sampleSelection", # Heckman type sample selection
            "censReg", # Censored Regression models
            "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)
```

Binary dependent variables are frequently studied in applied economics. Because a dummy variable y can only take values 0 and 1, its (conditional) expected value is equal to the (conditional) probability that $y = 1$:

$E(y|x) = 0 \times P(y = 0|x) + 1 \times P(y = 1|x) = P(y = 1|x)$

An important class of models specifies the success probability as

$P(y = 1|x) = G(\beta_0 + \beta_1 + \dots \beta_k x_k) = G(\boldsymbol{x\beta})$

The following table is taken from **?** and summarizes three examples of a generalized linear model:

	Linear Regression	Poisson Regression	Logistic Regression
$Y \mid \mathbf{X} = \mathbf{x}$	$N(\mu(\mathbf{x}), \sigma^2)$	$\text{Pois}(\lambda(\mathbf{x}))$	$\text{Bern}(p(\mathbf{x}))$
Distribution Name	Normal	Poisson	Bernoulli (Binomial)
$\text{E}[Y \mid \mathbf{X} = \mathbf{x}]$	$\mu(\mathbf{x})$	$\lambda(\mathbf{x})$	$p(\mathbf{x})$
Support	Real: $(-\infty, \infty)$	Integer: $0, 1, 2, \ldots$	Integer: $0, 1$
Usage	Numeric Data	Count (Integer) Data	Binary (Class) Data
Link Name	Identity	Log	Logit
Link Function	$\eta(\mathbf{x}) = \mu(\mathbf{x})$	$\eta(\mathbf{x}) = \log(\lambda(\mathbf{x}))$	$\eta(\mathbf{x}) = \log\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right)$
Mean Function	$\mu(\mathbf{x}) = \eta(\mathbf{x})$	$\lambda(\mathbf{x}) = e^{\eta(\mathbf{x})}$	$p(\mathbf{x}) = \frac{e^{\eta(\mathbf{x})}}{1+e^{\eta(\mathbf{x})}} = \frac{1}{1+e^{-\eta(\mathbf{x})}}$

Like ordinary linear regression, we will seek to "fit" the model by estimating the β parameters. To do so, we will use the method of maximum likelihood.

Note that a Bernoulli distribution is a specific case of a binomial distribution where the n parameter of a binomial is 1. Binomial regression is also possible, but we'll focus on the much more popular Bernoulli case.

So, in general, GLMs relate the mean of the response to a linear combination of the predictors, $\eta(\mathbf{x})$, through the use of a link function, $g()$. That is,

$$\eta(\mathbf{x}) = g\left(\text{E}[Y \mid \mathbf{X} = \mathbf{x}]\right).$$

The mean is then

$$\text{E}[Y \mid \mathbf{X} = \mathbf{x}] = g^{-1}(\eta(\mathbf{x})).$$

2.1 Linear probability models

If a dummy variable is used as the dependent variable y, we can still use OLS to estimate its relation to the regressors x.

```r
data("mroz", package = "wooldridge")
# Estimate linear probability model
linprob <- lm(inlf~nwifeinc+educ+exper+I(exper^2)+age+kidslt6+kidsge6,data=mroz)
# Regression table with heteroscedasticity-robust SE and t tests:
coeftest(linprob,vcov=hccm)
```

```
##
## t test of coefficients:
##
##                Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)  0.58551922  0.15358032  3.8125  0.000149 ***
## nwifeinc    -0.00340517  0.00155826 -2.1852  0.029182 *
## educ         0.03799530  0.00733982  5.1766 2.909e-07 ***
## exper        0.03949239  0.00598359  6.6001 7.800e-11 ***
## I(exper^2)  -0.00059631  0.00019895 -2.9973  0.002814 **
## age         -0.01609081  0.00241459 -6.6640 5.183e-11 ***
## kidslt6     -0.26181047  0.03215160 -8.1430 1.621e-15 ***
## kidsge6      0.01301223  0.01366031  0.9526  0.341123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimated coefficient *educ* can be interpreted as: an additional year of schooling increases the probability that a woman is in the labor force *ceteris paribus* by 3.80%.

One problem with linear probability models is that $P(y = 1|x)$ is specified as a linear function of the regressors. By construction, there are more or less realistic combinations of regressor values that yield $\hat{y} < 0$ or $\hat{y} > 1$.

2.2 Logit and Probit Models: Estimation

For binary response models, the most widely used specification for G are

- the **probit** model with $G(z) = \phi(z)$, the standard cdf and
- the **logit** model with $G(z) = \Lambda(z) = \frac{exp(z)}{1+exp(z)}$, the cdf of the logistic distribution.

In R many generalized linear models can be estimated by the `glm()` command. It accepts the additional option

- `family = binomial (link = logit)` for the logit model or
- `family = binomial (link = probit)` for the probit model

```
# Estimate logit model
logitres<-glm(inlf~nwifeinc+educ+exper+I(exper^2)+age+kidslt6+kidsge6,
              family=binomial(link=logit),data=mroz)
# Summary of results:
summary(logitres)
```

```
##
## Call:
## glm(formula = inlf ~ nwifeinc + educ + exper + I(exper^2) + age +
##     kidslt6 + kidsge6, family = binomial(link = logit), data = mroz)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1770  -0.9063   0.4473   0.8561   2.4032
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.425452   0.860365   0.495  0.62095
## nwifeinc    -0.021345   0.008421  -2.535  0.01126 *
## educ         0.221170   0.043439   5.091 3.55e-07 ***
## exper        0.205870   0.032057   6.422 1.34e-10 ***
## I(exper^2)  -0.003154   0.001016  -3.104  0.00191 **
## age         -0.088024   0.014573  -6.040 1.54e-09 ***
## kidslt6     -1.443354   0.203583  -7.090 1.34e-12 ***
## kidsge6      0.060112   0.074789   0.804  0.42154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.75  on 752  degrees of freedom
## Residual deviance:  803.53  on 745  degrees of freedom
## AIC: 819.53
##
## Number of Fisher Scoring iterations: 4
```

```
# Log likelihood value:
logLik(logitres)
```

```
## 'log Lik.' -401.7652 (df=8)
```

```
# McFadden's pseudo R2:
1 - logitres$deviance/logitres$null.deviance
```

```
## [1] 0.2196814
```

2.2.1 Multinomial Logit

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

can take one of three values:

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

```
data("nels_small", package="PoEdata")
```

```
nels.multinom <- multinom(psechoice~grades, data=nels_small)
```

```
## # weights:  9 (4 variable)
## initial  value 1098.612289
## iter  10 value 875.313116
## final  value 875.313099
## converged
```

```
summary(nels.multinom)
```

```
## Call:
## multinom(formula = psechoice ~ grades, data = nels_small)
##
## Coefficients:
##   (Intercept)     grades
## 2    2.505273 -0.3086404
## 3    5.770170 -0.7062468
##
## Std. Errors:
##   (Intercept)     grades
## 2   0.4183944 0.05228532
## 3   0.4043290 0.05292638
##
## Residual Deviance: 1750.626
## AIC: 1758.626
```

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

```
medGrades <- median(nels_small$grades)
fifthPercentileGrades <- quantile(nels_small$grades, .05)
newdat <- data.frame(grades=c(medGrades, fifthPercentileGrades))
```

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

```
##             1          2         3
##    0.18101808 0.28557312 0.5334088
## 5% 0.01781764 0.09662199 0.8855604
```

2.2.2 The Conditional Logit Model

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

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

- pepsi
- sevenup
- coke

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

```r
data("cola", package="PoEdata")
N <- nrow(cola)
N3 <- N/3
price1 <- cola$price[seq(1,N,by=3)]
price2 <- cola$price[seq(2,N,by=3)]
price3 <- cola$price[seq(3,N,by=3)]

bchoice <- rep("1", N3)
for (j in 1:N3){
  if(cola$choice[3*j-1]==1) bchoice[j] <- "2"
  if(cola$choice[3*j]==1) bchoice[j] <- "3"
}
cola.clogit <- MCMCmnl(bchoice ~
                         choicevar(price1, "b2", "1")+
                         choicevar(price2, "b2", "2")+
                         choicevar(price3, "b2", "3"),
                       baseline="3", mcmc.method="IndMH")
```

```
## Calculating MLEs and large sample var-cov matrix.
## This may take a moment...
## Inverting Hessian to get large sample var-cov matrix.
```

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

Table 2.2: Conditional logit estimates for the 'cola' problem

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

Table 2.3: Ordered probit estimates for the 'nels' problem

	Mean	SD
(Intercept)	2.9542	0.1478
grades	-0.3074	0.0193
gamma2	0.8616	0.0487

2.2.3 Ordered Choice Models

The order of choices in these models is meaningful, unlike the multinomial and conditional logit model we have studied so far. The following example explains the choice of higher education, when the choice variable is psechoice and the only regressor is grades; the dataset, *nels_small* , is already known to us.

The R package **MCMCpack** is again used here, with its function `MCMCoprobit()`.

```
nels.oprobit <- MCMCoprobit(psechoice ~ grades,
                            data=nels_small, mcmc=10000)
sOprobit <- summary(nels.oprobit)
tabOprobit <- sOprobit$statistics[, 1:2]
kable(tabOprobit, digits=4, align="c",
      caption="Ordered probit estimates for the 'nels' problem")
```

The results from MCMCoprobit can be translated into the textbook notations as follows:

- $\mu_1 = -$ (Intercept)
- $\beta =$ grades
- $\mu_2 =$ gamma2 $-$ (Intercept)

The probabilities for each choice can be calculated as in the next code fragment:

```
mu1 <- -tabOprobit[1]
b <- tabOprobit[2]
mu2 <- tabOprobit[3]-tabOprobit[1]
xGrade <- c(mean(nels_small$grades),
            quantile(nels_small$grades, 0.05))

# Probabilities:
prob1 <- pnorm(mu1-b*xGrade)
prob2 <- pnorm(mu2-b*xGrade)-pnorm(mu1-b*xGrade)
prob3 <- 1-pnorm(mu2-b*xGrade)

# Marginal effects:
Dp1DGrades <- -pnorm(mu1-b*xGrade)*b
Dp2DGrades <- (pnorm(mu1-b*xGrade)-pnorm(mu2-b*xGrade))*b
Dp3DGrades <- pnorm(mu2-b*xGrade)*b
```

For instance, the marginal effect of grades on the probability of attending a four-year college for a student with average grade and for a student in the top 5 percent are, respectively, -0.143 and -0.031.

2.2.4 Probit model

```
# Estimate probit model
probitres<-glm(inlf~nwifeinc+educ+exper+I(exper^2)+age+kidslt6+kidsge6,
               family=binomial(link=probit),data=mroz)
# Summary of results:
summary(probitres)
```

```
##
## Call:
## glm(formula = inlf ~ nwifeinc + educ + exper + I(exper^2) + age +
##     kidslt6 + kidsge6, family = binomial(link = probit), data = mroz)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2156  -0.9151   0.4315   0.8653   2.4553
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.2700736  0.5080782   0.532  0.59503
## nwifeinc    -0.0120236  0.0049392  -2.434  0.01492 *
## educ         0.1309040  0.0253987   5.154 2.55e-07 ***
## exper        0.1233472  0.0187587   6.575 4.85e-11 ***
## I(exper^2)  -0.0018871  0.0005999  -3.145  0.00166 **
## age         -0.0528524  0.0084624  -6.246 4.22e-10 ***
## kidslt6     -0.8683247  0.1183773  -7.335 2.21e-13 ***
## kidsge6      0.0360056  0.0440303   0.818  0.41350
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1029.7  on 752  degrees of freedom
## Residual deviance:  802.6  on 745  degrees of freedom
## AIC: 818.6
##
## Number of Fisher Scoring iterations: 4
```

```
# Log likelihood value:
logLik(probitres)
```

```
## 'log Lik.' -401.3022 (df=8)
```

```
# McFadden's pseudo R2:
1 - probitres$deviance/probitres$null.deviance
```

```
## [1] 0.2205805
```

2.2.5 Inference

We can implement the test for overall significance for the probit model using both manual and automatic calculations.

```
# Test of overall significance:
# Manual calculation of the LR test statistic:
```

```
probitres$null.deviance - probitres$deviance
```

```
## [1] 227.142
```

```
# Automatic calculations including p-values,...:
lrtest(probitres)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
8	-401			
1	-515	-7	227	2.01e-45

```
# Test of H0: experience and age are irrelevant
restr <- glm(inlf~nwifeinc+educ+ kidslt6+kidsge6,
             family=binomial(link=logit),data=mroz)
lrtest(restr,probitres)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
5	-465			
8	-401	3	127	2.12e-27

2.2.6 Predictions

The command `predict()` can calculate predicted values for the estimation sample or arbitrary sets of regressor values.

We can calculate

```
# predictions for two "extreme" women:
xpred <- list(nwifeinc=c(100,0),educ=c(5,17),exper=c(0,30),
              age=c(20,52),kidslt6=c(2,0),kidsge6=c(0,0))
# Predictions from linear probability, probit and logit model:
predict(linprob,  xpred,type = "response")
```

```
##          1          2
## -0.4104582  1.0428084
```

```
predict(logitres, xpred,type = "response")
```

```
##           1           2
## 0.005218002 0.950049117
```

```
predict(probitres,xpred,type = "response")
```

```
##           1           2
## 0.001065043 0.959869044
```

```
# Simulated data
set.seed(12345)
y <- rbinom(100, 1, 0.5)
x <- rnorm(100) + 2 * y

# Estimation
linpr.res <- lm(y~x)
```

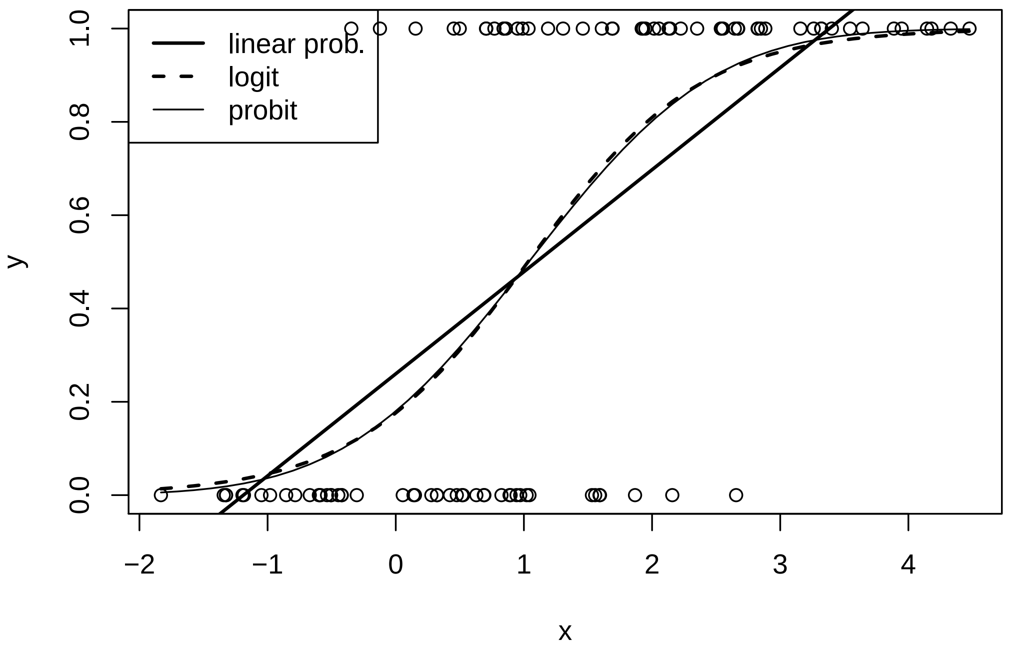


Figure 2.1: Predictions from binary response models (simulated data)

```r
logit.res <-glm(y ~x, family = binomial(link = logit))
probit.res <-glm(y ~x, family = binomial(link = probit))

# Prediction from a regular grid of x values
xp <- seq(from= min(x), to=max(x), length=50)
linpr.p <- predict(linpr.res, list(x = xp), type = "response")
logit.p <- predict(logit.res, list(x = xp), type = "response")
probit.p <- predict(probit.res, list(x = xp), type = "response")
```

```r
plot(x,y)
lines(xp, linpr.p, lwd=2, lty = 1)
lines(xp, logit.p, lwd=2, lty = 2)
lines(xp, probit.p, lwd=1, lty = 1)
legend("topleft",
       c("linear prob.", "logit", "probit"),
       lwd = c(2,2,1), lty = c(1,2,1))
```

2.2.7 Marginal (partial) effects

Several packages provide estimates of marginal effects for different types of models. Among these are **car**, **alr3**, **mfx**, **erer**, among others. Unfortunately, none of these packages implement marginal effects correctly (i.e., correctly account for interrelated variables such as interaction terms (e.g., a:b) or power terms (e.g., I(a^2)) and the packages all implement quite different interfaces for different types of models. The **margins** and **prediction** packages are a combined effort to calculate marginal effects that include complex terms and provide a uniform interface for doing those calculations.

To know how much a variable influences the labour force participation, one has to use `margins()` command:

```r
effects_logit_participation <- margins(logitres)
summary(effects_logit_participation)
```

factor	AME	SE	z	p	lower	upper
age	-0.0157	0.00238	-6.6	4.04e-11	-0.0204	-0.0111
educ	0.0395	0.00729	5.41	6.15e-08	0.0252	0.0538
exper	0.0254	0.00224	11.4	5.99e-30	0.021	0.0298
kidsge6	0.0107	0.0133	0.805	0.421	-0.0154	0.0369
kidslt6	-0.258	0.0319	-8.07	7.05e-16	-0.32	-0.195
nwifeinc	-0.00381	0.00148	-2.57	0.0101	-0.00672	-0.000906

If one desires subgroup effects, simply pass a subset of data to the data argument:

```
summary(margins(logitres, data = subset(mroz, kidslt6 == 0))) # no kids < 6 years
```

factor	AME	SE	z	p	lower	upper
age	-0.0156	0.00234	-6.64	3.04e-11	-0.0202	-0.011
educ	0.0391	0.00726	5.39	7.13e-08	0.0249	0.0534
exper	0.0246	0.00212	11.6	4.23e-31	0.0204	0.0287
kidsge6	0.0106	0.0132	0.807	0.42	-0.0152	0.0365
kidslt6	-0.255	0.033	-7.73	1.09e-14	-0.32	-0.191
nwifeinc	-0.00378	0.00147	-2.57	0.0102	-0.00666	-0.000894

```
ggplot(data = summary(effects_logit_participation)) +
  geom_point(aes(factor, AME)) +
  geom_errorbar(aes(x = factor, ymin = lower, ymax = upper)) +
  geom_hline(yintercept = 0, color="lightgrey") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45))
```

You can also extract the marginal effects of a single variable, with `dydx`:

```
head(dydx(mroz, logitres, "educ"))
```

dydx_educ
0.0464
0.0416
0.0463
0.0384
0.0538
0.0335

The function `cplot()` provides the commonly needed visual summaries of predictions or average marginal effects conditional on a covariate.

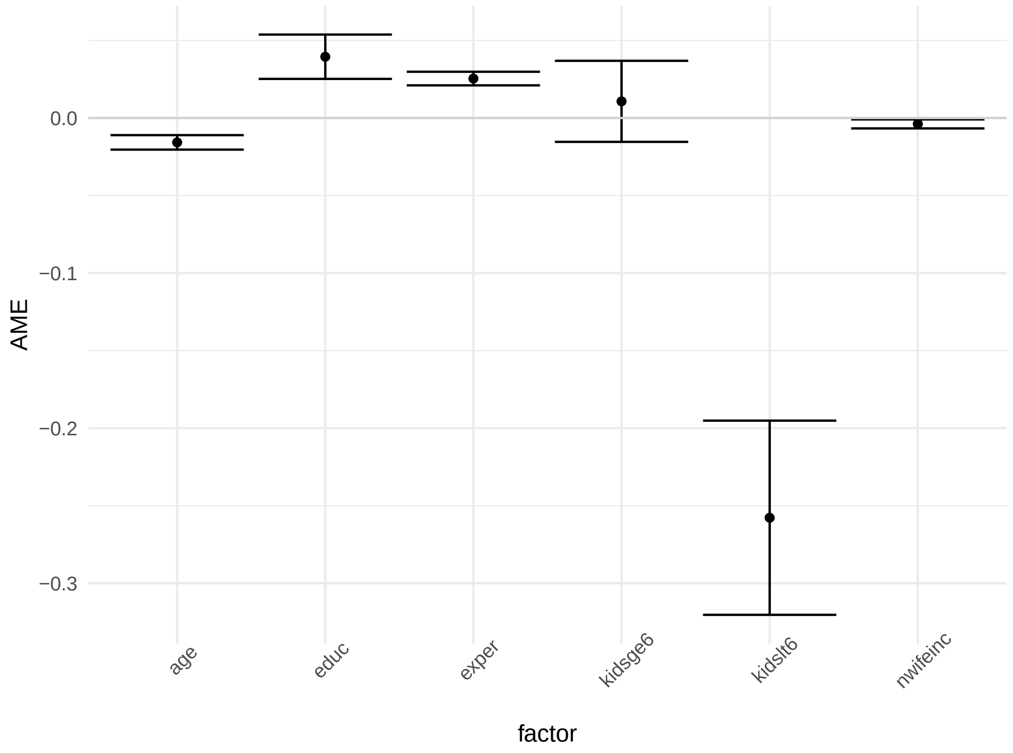


Figure 2.2: Logit effect plot

```r
cplot(logitres, x = "educ", se.type = "shade")
```

```
##    xvals     yvals     upper     lower
## 1    5.0 0.2549509 0.3787666 0.1311353
## 2    5.5 0.2765192 0.3988913 0.1541471
## 3    6.0 0.2991794 0.4190899 0.1792689
## 4    6.5 0.3228678 0.4392936 0.2064421
## 5    7.0 0.3475019 0.4594481 0.2355556
## 6    7.5 0.3729801 0.4795186 0.2664417
## 7    8.0 0.3991836 0.4994946 0.2988726
## 8    8.5 0.4259774 0.5193953 0.3325594
## 9    9.0 0.4532129 0.5392750 0.3671508
## 10   9.5 0.4807315 0.5592299 0.4022331
## 11  10.0 0.5083674 0.5794045 0.4373304
## 12  10.5 0.5359524 0.5999963 0.4719084
## 13  11.0 0.5633190 0.6212490 0.5053891
## 14  11.5 0.5903055 0.6434188 0.5371922
## 15  12.0 0.6167588 0.6666929 0.5668248
## 16  12.5 0.6425384 0.6910722 0.5940047
## 17  13.0 0.6675187 0.7162931 0.6187444
## 18  13.5 0.6915913 0.7418704 0.6413122
## 19  14.0 0.7146659 0.7672368 0.6620950
## 20  14.5 0.7366715 0.7918749 0.6814681
```

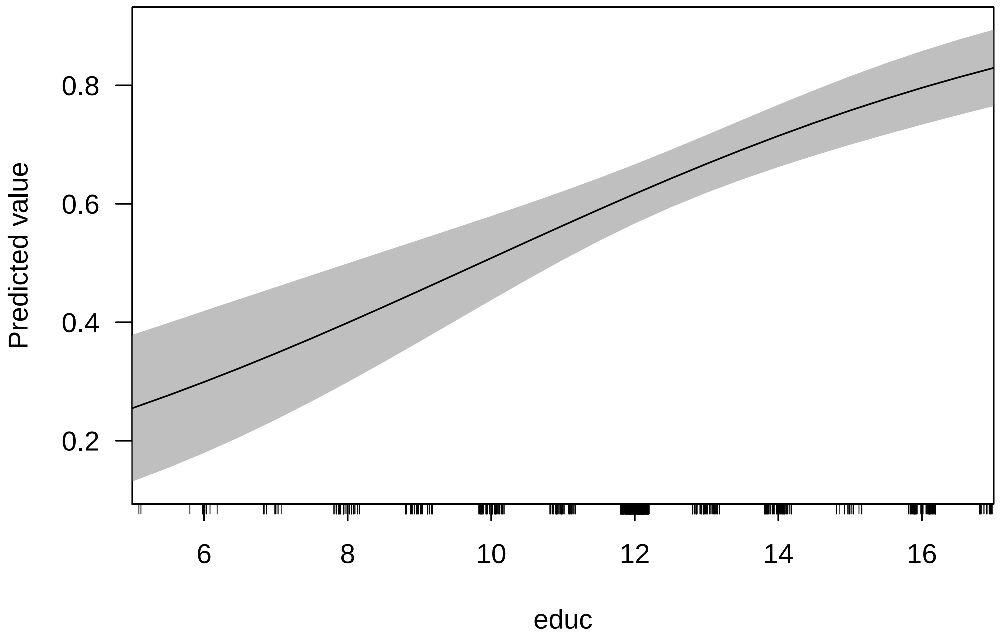


Figure 2.3: Marginal effects for logit model

2.3 Count data: The Poisson Regression Model

Instead of just 0/1-coded binary data, count data can take any non-negative integer $0, 1, 2, \ldots$. If they take very large numbers (like the number of students in a school), they can be approximated reasonably well as contiouns variables in a linear models and estimated using OLS. If the numbers are relativly small, this approximation might not work well.

The Poisson regression model is the most basic and convenient model explicitly model explicitly designed for count data.

Poisson regression models can be estimated in R via the `glm()` function with the specification **family= poisson**

Estimating the model with **quasipoisson** is to adjust for potential vioalations of the Poisson distribution.

```
data(crime1, package='wooldridge')

# Estimate linear model
lm.res      <-  lm(narr86~pcnv+avgsen+tottime+ptime86+qemp86+inc86+
                     black+hispan+born60, data=crime1)
# Estimate Poisson model
Poisson.res <- glm(narr86~pcnv+avgsen+tottime+ptime86+qemp86+inc86+
                     black+hispan+born60, data=crime1, family=poisson)
# Quasi-Poisson model
QPoisson.res<- glm(narr86~pcnv+avgsen+tottime+ptime86+qemp86+inc86+
                     black+hispan+born60, data=crime1, family=quasipoisson)
```

```
stargazer(lm.res,Poisson.res,QPoisson.res,type="text",keep.stat="n")
```

```
##
## ====================================================
##                       Dependent variable:
##              --------------------------------------
##                              narr86
##                  OLS     Poisson  glm: quasipoisson
##                                      link = log
##                  (1)       (2)           (3)
## ---------------------------------------------------
## pcnv          -0.132*** -0.402***     -0.402***
##                (0.040)   (0.085)       (0.105)
##
## avgsen         -0.011    -0.024        -0.024
##                (0.012)   (0.020)       (0.025)
##
## tottime         0.012    0.024*         0.024
##                (0.009)   (0.015)       (0.018)
##
## ptime86       -0.041*** -0.099***     -0.099***
##                (0.009)   (0.021)       (0.025)
##
## qemp86        -0.051***  -0.038        -0.038
##                (0.014)   (0.029)       (0.036)
##
## inc86         -0.001*** -0.008***     -0.008***
##               (0.0003)   (0.001)       (0.001)
##
## black         0.327***  0.661***      0.661***
##                (0.045)   (0.074)       (0.091)
##
## hispan        0.194***  0.500***      0.500***
##                (0.040)   (0.074)       (0.091)
##
## born60         -0.022    -0.051        -0.051
##                (0.033)   (0.064)       (0.079)
##
## Constant      0.577***  -0.600***     -0.600***
##                (0.038)   (0.067)       (0.083)
##
## ---------------------------------------------------
## Observations   2,725     2,725         2,725
## ===================================================
## Note:                  *p<0.1; **p<0.05; ***p<0.01
```

By construction, the parameter estiamtes are the same but the standard errors are larger for the QMLE.

2.3.1 Corner Solution Response: The Tobit Model

Corner solutions describe situations where the variable of interest is continous but restricted in range. Typically, it cannot be negative.

The package **censReg** offers the command `censReg()` for estimating a Tobit model.

We have already estimated labor supply mdoeld for the women in the dataset *mroz*, ignoring the fact that the hours worked is necessarily non-negative.

```
# Estimate Tobit model using censReg:
TobitRes <- censReg(hours~nwifeinc+educ+exper+I(exper^2)+
                      age+kidslt6+kidsge6, data=mroz )
```

```
summary(TobitRes)
```

```
##
## Call:
## censReg(formula = hours ~ nwifeinc + educ + exper + I(exper^2) +
##     age + kidslt6 + kidsge6, data = mroz)
##
## Observations:
##          Total  Left-censored     Uncensored Right-censored
##            753            325            428              0
##
## Coefficients:
##                Estimate Std. error t value  Pr(> t)
## (Intercept)  965.30528  446.43613   2.162 0.030599 *
## nwifeinc      -8.81424    4.45910  -1.977 0.048077 *
## educ          80.64561   21.58324   3.736 0.000187 ***
## exper        131.56430   17.27939   7.614 2.66e-14 ***
## I(exper^2)    -1.86416    0.53766  -3.467 0.000526 ***
## age          -54.40501    7.41850  -7.334 2.24e-13 ***
## kidslt6     -894.02174  111.87803  -7.991 1.34e-15 ***
## kidsge6      -16.21800   38.64139  -0.420 0.674701
## logSigma       7.02289    0.03706 189.514  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Newton-Raphson maximisation, 7 iterations
## Return code 1: gradient close to zero
## Log-likelihood: -3819.095 on 9 Df
```

```
# Partial Effects at the average x:
margEff(TobitRes)
```

```
##    nwifeinc        educ       exper  I(exper^2)         age     kidslt6
##   -5.326442   48.734094   79.504232   -1.126509  -32.876918 -540.256831
##     kidsge6
##   -9.800526
```

Another alternative for estimating Tobit models is the command **survreg** from the package survival. It is less straightforward to use but more flexible.

```
# Estimate Tobit model using survreg:
res <- survreg(Surv(hours, hours>0, type="left") ~ nwifeinc+educ+exper+
                 I(exper^2)+age+kidslt6+kidsge6, data=mroz, dist="gaussian")
summary(res)
```

```
##
## Call:
## survreg(formula = Surv(hours, hours > 0, type = "left") ~ nwifeinc +
##     educ + exper + I(exper^2) + age + kidslt6 + kidsge6, data = mroz,
##     dist = "gaussian")
##                  Value Std. Error      z       p
## (Intercept)  965.3053   446.4361   2.16 0.03060
## nwifeinc      -8.8142     4.4591  -1.98 0.04808
## educ          80.6456    21.5832   3.74 0.00019
## exper        131.5643    17.2794   7.61 2.7e-14
## I(exper^2)    -1.8642     0.5377  -3.47 0.00053
## age          -54.4050     7.4185  -7.33 2.2e-13
## kidslt6     -894.0217   111.8780  -7.99 1.3e-15
## kidsge6      -16.2180    38.6414  -0.42 0.67470
## Log(scale)     7.0229     0.0371 189.51 < 2e-16
##
## Scale= 1122
##
## Gaussian distribution
## Loglik(model)= -3819.1   Loglik(intercept only)= -3954.9
##  Chisq= 271.59 on 7 degrees of freedom, p= 7e-55
## Number of Newton-Raphson Iterations: 4
## n= 753
```

2.4 Censored and Truncated Regression Models

Censored regression models are closely related to Tobit models. In the basic Tobit model we observe $y = y^*$ in the "uncensored" cases with $y^* > 0$ and we only know an upper bound for $y^* \leq 0$ if we observe $y 0 = $.

In this example we are are interested in criminal prognosis of individuals released from prison to reoffend.

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

# Define Dummy for UNcensored observations
recid$uncensored <- recid$cens==0
# Estimate censored regression model:
res<-survreg(Surv(log(durat),uncensored, type="right") ~ workprg+priors+
               tserved+felon+alcohol+drugs+black+married+educ+age,
             data=recid, dist="gaussian")
# Output:
summary(res)
```

```
##
## Call:
## survreg(formula = Surv(log(durat), uncensored, type = "right") ~
##     workprg + priors + tserved + felon + alcohol + drugs + black +
##         married + educ + age, data = recid, dist = "gaussian")
##                 Value Std. Error     z       p
## (Intercept)  4.099386   0.347535 11.80 < 2e-16
## workprg     -0.062572   0.120037 -0.52  0.6022
## priors      -0.137253   0.021459 -6.40 1.6e-10
## tserved     -0.019331   0.002978 -6.49 8.5e-11
## felon        0.443995   0.145087  3.06  0.0022
## alcohol     -0.634909   0.144217 -4.40 1.1e-05
## drugs       -0.298160   0.132736 -2.25  0.0247
## black       -0.542718   0.117443 -4.62 3.8e-06
## married      0.340684   0.139843  2.44  0.0148
## educ         0.022920   0.025397  0.90  0.3668
## age          0.003910   0.000606  6.45 1.1e-10
## Log(scale)   0.593586   0.034412 17.25 < 2e-16
##
## Scale= 1.81
##
## Gaussian distribution
## Loglik(model)= -1597.1   Loglik(intercept only)= -1680.4
##  Chisq= 166.74 on 10 degrees of freedom, p= 1.3e-30
## Number of Newton-Raphson Iterations: 4
## n= 1445
```

2.4.1 The Heckman, or Sample Selection Model

The models are useful when the sample selection is not random, but whehter an individual is in the sample depends on individual characteristics. For example, when studying wage determination for married women, some women are not in the labour force, therefore their wages are zero.

The Heckit procedure involves two steps, estimating both the selection equation and the equation of interest. Function `selection()` in the **sampleSelection** package performs both steps; therefore, it needs both equations among its arguments. (The selection equation is, in fact, a probit model.)

```
data("mroz", package='wooldridge')

wage.heckit <- selection(inlf~age+educ+I(kidslt6+kidsge6)+mtr,
                         log(wage)~educ+exper,
                         data=mroz, method="ml")
summary(wage.heckit)
```

```
## --------------------------------------------
## Tobit 2 model (sample selection model)
## Maximum Likelihood estimation
## Newton-Raphson maximisation, 4 iterations
## Return code 2: successive function values within tolerance limit
## Log-Likelihood: -913.5131
## 753 observations (325 censored and 428 observed)
## 10 free parameters (df = 743)
## Probit selection equation:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            1.53798    0.61889   2.485   0.0132 *
## age                   -0.01346    0.00603  -2.232   0.0259 *
## educ                   0.06278    0.02180   2.879   0.0041 **
## I(kidslt6 + kidsge6) -0.05108    0.03276  -1.559   0.1194
## mtr                   -2.20864    0.54620  -4.044 5.81e-05 ***
## Outcome equation:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.646221   0.235569   2.743  0.00623 **
## educ        0.066462   0.016573   4.010 6.68e-05 ***
## exper       0.011972   0.004085   2.931  0.00348 **
##    Error terms:
##       Estimate Std. Error t value Pr(>|t|)
## sigma  0.84112    0.04302   19.55   <2e-16 ***
## rho   -0.82768    0.03911  -21.16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## --------------------------------------------
```

Bibliography

Adkins, L. Using gretl for Principles of Econometrics, 4th Edition.

Dalpiaz, D. *Applied Statistics*.

Heiss, F. (2016). *Using R for Introductory Econometrics*. Florian Heiss.

Hill, R. C., Griffiths, W. E., Lim, G. C., and Lim, M. A. (2008). *Principles of econometrics*, volume 5. Wiley Hoboken, NJ.

Oswald, F. and Robin, J.-M. (2018). *Introduction to Econometrics with R*. github.com.

Rodrigues, B. Econometrics and Free Software.

Team, R. C. (2013). R: A language and environment for statistical computing [internet]. *Vienna, Austria: R foundation for statistical computing*.

Wooldridge, J. M. (2015). *Introductory econometrics: A modern approach*. Nelson Education.

Zeileis, A., Lumley, T., Berger, S., and Graham, N. Sandwich: Robust Covariance Matrix Estimators.