Data and statistics with R

Jüri Lillemets

2019-08-30

Contents

Preface

Purposes of the course: quantitative data management, processing and analysis with R, tidying, description, visualization, statistical analysis.

Getting help: ? command, Stackoverflow

> R is a **language** and **environment** for **statistical computing** and **graphics**.

Based on S language orginating from 1970's. Developed during 1990's and became public around 2000.

Language and environment. Programming language similar to any other but developed paricularly for data analysis. Flexible and extensible environment as opposed to many statistical packages. Command line interface.

Statistical computing and graphics. Includes many statistical procedures for various fields. Constantly extended by the community with novel methods. Abundant possibilities for plotting data

Free and open source as opposed to most statistical packages. Powerful. Allows for reproducibile analysis. Extensible. Active community.

Command line interface

```
R version 3.5.3 (2019-03-11) -- "Great Truth"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Apply functions on objects. Data is not constantly visible. No undo. Multiple ways to get the same result.

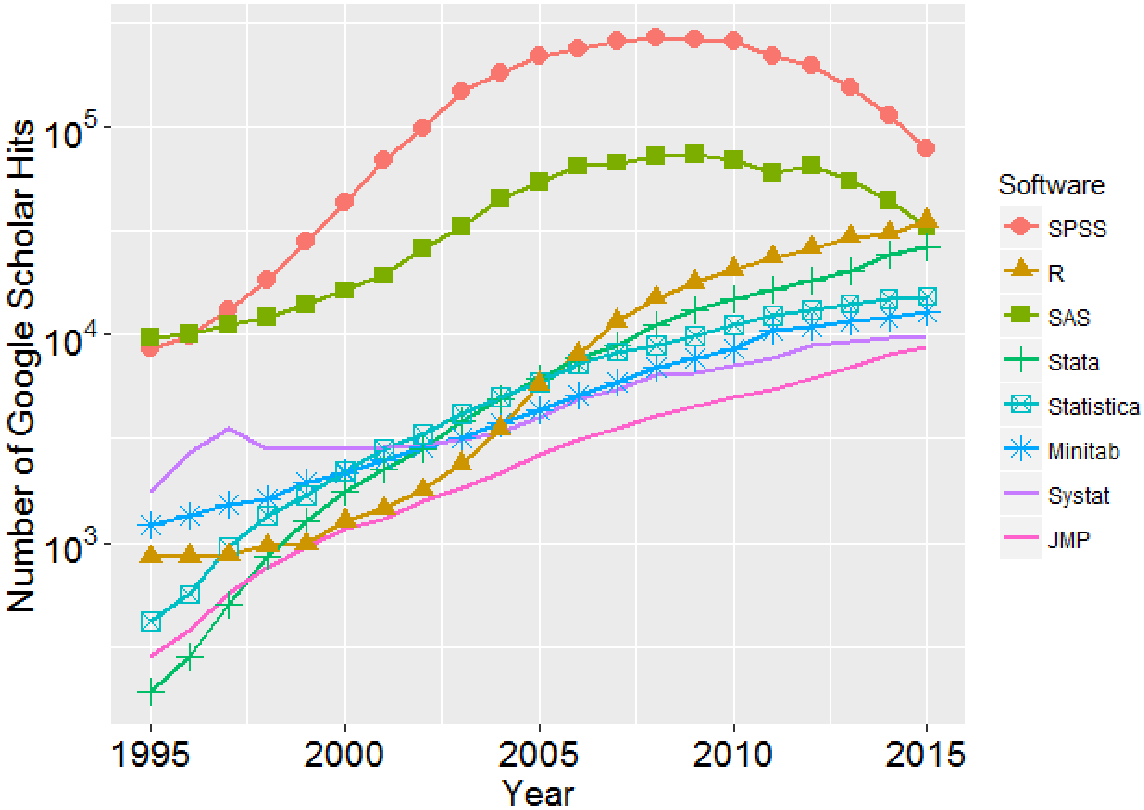


Figure 1: http://r4stats.com/2016/06/08/r-passes-sas-in-scholarly-use-finally/

Chapter 1

Basic data management in R

1.1 Basic R use

1.1.1 Arithmetics

R is a functional programming language. One of the implications of this is that R evaluates expressions and returns the result instead of assigning values to variables through statements. This functionality of R includes basic arithmetic functions.

```
5 + 2 # Addition
5 - 2 # Subtraction
5 * 2 # Multiplication
5 / 2 # Division
5 %% 2 # Return only the remainder of division
5 %/% 2 # Return the result of division without remainder
```

Some more advanced mathematical expressions can also be evaluated using R. Most of these require input to be explicitly passed to functions instead of using operators as in the previous example.

```
5 ^ 2 # 5 to the power of 2
sqrt(25) # Square root of 25
exp(1) # Exponent
log(5) # Natural logarithm
```

Finally, the order of operations can be set with parenthesis.

```
5 - 2 * 3
```

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

```
5 - (2 * 3)
```

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

Thus, for users comfortable with typing, R can also function as a very handy and advanced calculator.

1.1.2 Relational operators

For the comparison of two objects, relational operators are used in R.

```
5 > 2 # 5 is greater than 2
5 < 2 # 5 is less than 2
2 >= 2 # 2 is equal to or greater than 2
2 >= 5 # 5 is equal to or less than 2
5 == 2 # 5 is equal to 2
2 != 2 # 2 is not equal to 2
```

If the sides of the operator contain uneven number of elements, these will be recycled.

```
5 > c(2, 5) # 5 is greater than 2, 5 is greater than 5
```

Naturally, all of these operators also work on character objects.

```
"String" == "String"
"String" != "String"
```

A very useful operator `%in%` can also be considered as a relational operator, evaluating whether or not an element is also included in another object.

```
5 %in% c(2, 5) # 5 is an element of a vector containing 2 and
"String" %in% c(2, 5)
```

1.1.3 Logical operators

Boolean logic is useful to compare objects of type logical (i.e. `TRUE` and `FALSE`)

```
TRUE | FALSE # True or false is true
TRUE & FALSE # True and false are both true
!TRUE # The opposite of true
```

Of course, these operators are not very useful for comparison of logical objects explicitly but are helpful together with expressions.

```
5 == 2 | 5 > 2 # 5 equals 2 or 5 is larger than 2
```

1.1.4 Assignment

Data in R is stored in objects. In most cases, values are assigned to objects by assignment operators. In a lot of programming languages, `=` is used for assignment. This works in R, too (which is why `==` must not be confused with `=`). However, for historical reasons it is customary to use `<-` for assignment in R. Object names can contain numbers, letters and punctuation marks but never start with a number.

```
a <- 5
a
```

```
## [1] 5
```

```
a = 5
a
```

```
## [1] 5
```

Note that the assignment operator must always be enclosed with spaces, otherwise R might evaluate the assignment as a comparison.

```
a <- 5 # Assign 5 to a
a<-5 # a is greater than -5
```

Another way to assign values is to use the `assign()` function but this usually better to be avoided.

1.1.5 Set operations

Although rarely necessary, operations on sets can be helpful to return specific elements from vectors.

```
a <- c(2, 2, 5, "String")
b <- c(1:3)
union(a, b) # All elements of a and b
intersect(a, b) # Elements in both a and b
setdiff(a, b) # Elements in a but not in b
setdiff(a, b) # a and b contain the same elements
```

While `union()` and `setdiff()` may seem to be the same as `c()` and `identical()` respectively, the set functions presented here behave differenly because they ignore duplications and ordering of elements.

1.2 Data structures

1.2.1 Levels of measurement

Traditional (S. S. Stevens)

- Nominal
- Ordinal
- Interval
- Ratio

Conventional scales

- Categorical/qualitative
  - Binary/dichotomous/boolean/logical
  - Nominal
  - Ordinal/ranked
- Numeric/quantitative
  - Continuous
  - Discrete
    - Interval
    - Ratio

In R

- Logical
- Character
- (Factor)
- Integer
- Double

Missing values in R

- NA - Not available/applicable
- NaN - Not a number
- Inf - positive infinite
- -Inf - Negative infinite

1.2.2 Types of objects in R

Homogenous	Heterogenous
Vector, c()	List, list()
Matrix, matrix()	Data frame, data.frame()
Array, array()	

To determine type and class of an object: `typeof()`, `class()`, `is.*...()*`

To treat object as if it was of a certain type: `as.*...()*`.

1.2.3 Subsetting

Vectors

```
a <- letters[1:12]
a[2]
```

```
## [1] "b"
```

```
a[-2]
```

```
##  [1] "a" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
```

```r
a[c(1, 3, 5)]
```

```
## [1] "a" "c" "e"
```

Lists

```r
a <- list(Letters = letters[1:12], Numbers = 1:10)
a$Letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
```

```r
a[1]
```

```
## $Letters
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
```

```r
a[[1]]
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
```

```r
a[[1]][1]
```

```
## [1] "a"
```

Matrices

```r
(a <- matrix(1:9, 3, 3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```r
a[1]
```

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

```r
a[1, ]
```

```
## [1] 1 4 7
```

```r
a[, 1]
```

```
## [1] 1 2 3
```

Data.frames

```r
mtcars$mpg
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

```r
mtcars[, 1]
```

```
##  [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3 15.2
## [15] 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3 26.0 30.4
## [29] 15.8 19.7 15.0 21.4
```

```r
mtcars[1:10, ]
```

```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
```

```
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

Using logical operators.

```
mtcars$mpg > 20
mtcars[mtcars$mpg > 20, ]
```

Using indexes.

```
order(mtcars$mpg)
```

```
##  [1] 15 16 24  7 17 31 14 23 22 29 12 13 11  6  5 10 25 30  1  2  4 32 21
## [24]  3  9  8 27 26 19 28 18 20
```

```
mtcars[order(mtcars$mpg), ]
```

```
##                      mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8 472.0 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3    4
## Camaro Z28          13.3   8 350.0 245 3.73 3.840 15.41  0  0    3    4
## Duster 360          14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Chrysler Imperial   14.7   8 440.0 230 3.23 5.345 17.42  0  0    3    4
## Maserati Bora       15.0   8 301.0 335 3.54 3.570 14.60  0  1    5    8
## Merc 450SLC         15.2   8 275.8 180 3.07 3.780 18.00  0  0    3    3
## AMC Javelin         15.2   8 304.0 150 3.15 3.435 17.30  0  0    3    2
## Dodge Challenger    15.5   8 318.0 150 2.76 3.520 16.87  0  0    3    2
## Ford Pantera L      15.8   8 351.0 264 4.22 3.170 14.50  0  1    5    4
## Merc 450SE          16.4   8 275.8 180 3.07 4.070 17.40  0  0    3    3
## Merc 450SL          17.3   8 275.8 180 3.07 3.730 17.60  0  0    3    3
## Merc 280C           17.8   6 167.6 123 3.92 3.440 18.90  1  0    4    4
## Valiant             18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Hornet Sportabout   18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Merc 280            19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
## Pontiac Firebird    19.2   8 400.0 175 3.08 3.845 17.05  0  0    3    2
## Ferrari Dino        19.7   6 145.0 175 3.62 2.770 15.50  0  1    5    6
## Mazda RX4           21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag       21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Hornet 4 Drive      21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Volvo 142E          21.4   4 121.0 109 4.11 2.780 18.60  1  1    4    2
## Toyota Corona       21.5   4 120.1  97 3.70 2.465 20.01  1  0    3    1
## Datsun 710          22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 230            22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 240D           24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Porsche 914-2       26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Fiat X1-9           27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Honda Civic         30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Lotus Europa        30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Fiat 128            32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Toyota Corolla      33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
```

`order()` returns indexes while `sort()` returns the elements.

1.3 Workspace management

To set working directory for the session: `setwd()`. On Windows filesystem, \ needs to be escaped (\\) or replaced with `/`. When working directory is set, all file paths must be relative to the specified directory. To return current working directory: `getwd()`.

To list all objects on workspace: `ls()`. To remove an object: `rm()`; and to remove all objects: `rm(list = ls())`.

1.4 Data sources and managing data

Natively R supports only plain text (e.g. `.csv`) and its native (`.Rdata` and `.Rds`) data formats. Most widely used plain text data format, the Comma-Separated Values (`.csv`) can be loaded by a dedicated function `read.csv()` by providing a location on disk or a url. For an alternative `.csv` where values are separated by semicolons is the `read.csv2()` function. For other plain text formats, `read.table()` allows to specify various attributes. When values are separated by tabs use `sep = \t`.

```
mtCars <- read.csv('some_file.csv`, stringsAsFactors = F)
```

When some data is used exclusively in R, the native R data formats should be used as these allow more efficient data compression. All objects currently on workspace can be saved with `save.image()` function and loaded afterwards using `load()`. In this case, `.Rdata` file should be used.

```
save.image('some_data.Rdata')
load('some_data.Rdata')
```

Sometimes it is necessary to only save a single object on workspace. Then the extension should be `.Rds` and the corresponding commands are `saveRDS` and `readRDS`.

```
saveRDS(a, 'some_data.Rds')
readRDS('some_data.Rds')
```

All common data formats native to other software can also be loaded into R, but this requires relevant libraries to be loaded. The `foreign` package contains functions to load data of other statistical packages, e.g. SAS, SPSS and Stata. For Excel formats, `readxl::read_excel` loads `.xlx` as well as `.xlsx` files while the `openxlsx` library provides functions to meticulously edit and save Excel workbooks.

1.4.1 Understanding a dataset

Once a dataset is loaded into R, it's a good idea to get an understanding of it. While an entire object can be viewed using `View()`, this is not feasible for anything but small tables. Instead, `structure()` (`str()`) displays an overview of all columns in a data frame, `names()` lists the names of all columns and `summary()` gives some statistics on the values of each column.

```
str(mtcars)
```

```
## 'data.frame':	32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

```
summary(mtcars)
```

```
##       mpg             cyl             disp             hp       
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0  
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
```

```
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear             carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000
```

```r
names(mtcars)
```

```
##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"
```

```r
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
tail(mtcars)
```

```
##                 mpg cyl  disp  hp drat    wt qsec vs am gear carb
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
## Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
## Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
## Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
## Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

1.5 Some R principles

1.5.1 Environments

R searches for objects in an environment where an operation is done. When it does not find an object there, it will incrementally search in higher environments.

```r
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```r
mtcars <- 1:10
head(mtcars)
```

```
## [1] 1 2 3 4 5 6
```

```
sum <- function(...) Reduce(`-`, ...)
sum(1, 10)
```

```
## [1] 9
```

1.5.2 Recycling

```
a <- 1:2
b <- 1:3
a + b
```

```
## Warning in a + b: longer object length is not a multiple of shorter object
## length
```

```
## [1] 2 4 4
```

1.5.3 Lazy evaluation

```
someFun <- function(x, y) print(x)
someFun(x = "Hello world!")
```

```
## [1] "Hello world!"
```

Chapter 2

R in practice

2.1 RStudio

Installation and interface

2.2 Data preparation and tidying

2.2.1 Wide and long format

reshape()

2.2.2 Combining

cbind, rbind

merge

2.3 Aggregation

aggregate

2.3.1 Duplication

unique, duplicated

2.3.2 String manipulation

substr, substring, grep, grepl, sub, gsub

xkcd: perl_problems

2.4 Functions

Brief introduction

Arguments etc.

2.5 Repetitive processing

For loops, the apply family

ifelse, for, lapply, sapply

Chapter 3

Extending R base functionality

3.1 Packages

3.1.1 CRAN

3.2 Tidyverse ecosystem

3.2.1 Split-apply-combine

dplyr::group_by()

3.2.2 Wide and long

tidyr::gather, tidyr::spread

Chapter 4

Descriptive statistics

4.1 Numbers

Absolute and relative.

xkcd: 1000_times

Percentage points.

xkcd: percentage_points

4.1.1 Normalizing and scaling

log_scale.png

4.2 Frequency tables

table, prop.table, aggregate, by

4.3 Measures of central tendency, quantiles...

mean, median, mode, sd, var, max, min, quantile

Chapter 5

Data visualization

5.1 Basic principles

Pie charts, 3 dimensions, y-scale, ...

scientific_paper_graph_quality.png

self_description.png

5.2 Types of plots

violin_plots.png

(scatter)plot, barplot, boxplot

pairs, biplot

histogram

5.3 Adding elements

Legend, text, abline

5.4 Parameters

mfrow, ...

5.5 Saving

5.6 Extending basic plotting

ggplot2

Chapter 6

Inferential statistics

6.1 Sample and population, distributions

6.2 Hypothesis testing

Parametric vs non-parametric tests.

Chapter 7

Measures of association

7.1 Contingency table

table()

7.1.1 Odds ratio

7.1.2 Chi2 test

7.2 Correlation

7.3 Goodness of fit and coeffiicent of determination

7.4 Regression

Chapter 8

Ordinary least squares regression

The requirements for a BLUE (**b**est **l**inear **u**nbiased **e**stimator) will be outlined.

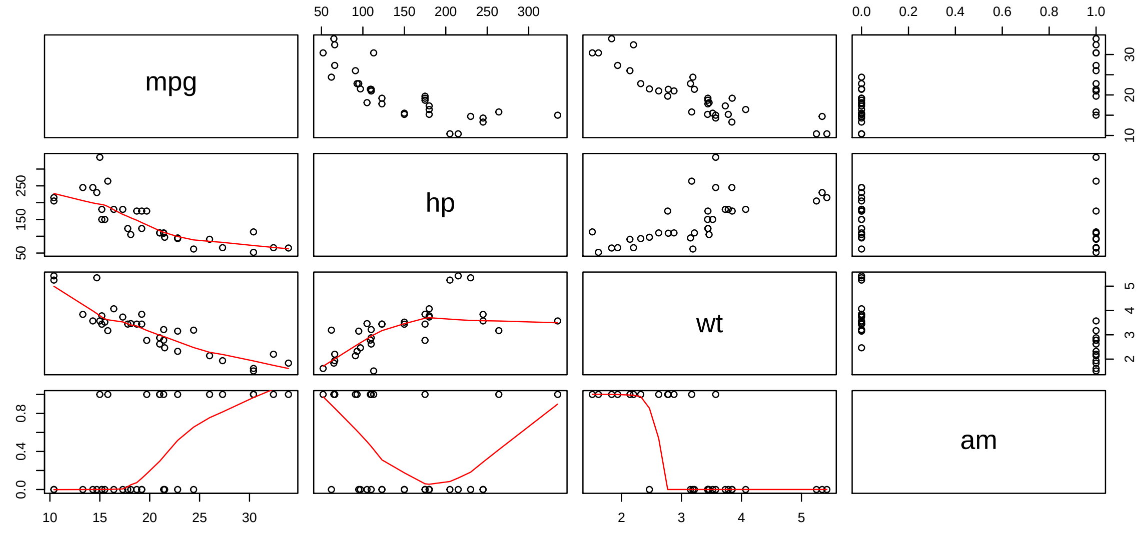
Dependent variables will be referred to as **response** and independent variable(s) as **predictor(s)**.

We'll be using the pipe operator for better readability so let's first load the `magrittr` package.

```
library(magrittr)
```

The `mtcars` dataset in base R is suitable for an example. To begin with, it's a good idea to plot the relationships between variables of interest together with a LOWESS smoothed line.

```
mtcars[, c('mpg', 'hp', 'wt', 'am')] %>% pairs(lower.panel = panel.smooth)
```



The relationships between variables are cleary present. Theoretically, we can expcet that horsepower (`hp`), weight (`wt`) and transmission (`am`, automatic (0) or manual (1)) influence fuel consumption (`mpg`). So let's model this relationship.

```
(mpgMod <- lm(mpg ~ hp + wt + factor(am), mtcars))
```

```
##
## Call:
## lm(formula = mpg ~ hp + wt + factor(am), data = mtcars)
##
## Coefficients:
## (Intercept)           hp           wt  factor(am)1
```

```
##    34.00288     -0.03748     -2.87858      2.08371
```

The resulting coefficients represent model parameters. We can also get 95% confidence intervals for these coefficients.

```r
confint(mpgMod)
```

```
##                    2.5 %      97.5 %
## (Intercept) 28.58963286 39.41611738
## hp          -0.05715454 -0.01780291
## wt          -4.73232353 -1.02482730
## factor(am)1 -0.73575874  4.90317900
```

A `summary` call of the model provides a lot of information that will be further explained.

```r
summary(mpgMod)
```

```
##
## Call:
## lm(formula = mpg ~ hp + wt + factor(am), data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4221 -1.7924 -0.3788  1.2249  5.5317
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 34.002875   2.642659  12.867 2.82e-13 ***
## hp          -0.037479   0.009605  -3.902 0.000546 ***
## wt          -2.878575   0.904971  -3.181 0.003574 **
## factor(am)1  2.083710   1.376420   1.514 0.141268
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.538 on 28 degrees of freedom
## Multiple R-squared:  0.8399, Adjusted R-squared:  0.8227
## F-statistic: 48.96 on 3 and 28 DF,  p-value: 2.908e-11
```

8.1 Model explains the data well

A model does not always have to explain the variation of response but in often this is useful, e.g. when comparing models. For linear least squares models goodness of fit can be measured by the coefficient of determination (R^2). Because it indicates the **part of variation that is explained by the model**, it can be represented by three measures:

- the sum of the squares of differences of each observed value and the mean value of response (**t**otal sum of squares, TSS), i.e. $\sum_{i=1}^{n}(y_i - \overline{y}_i)^2$
- the sum of the squares of differences of the predicted values and the mean value of response (**e**xplained sum of squares, ESS), i.e. $\sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$
- the sum of the squares of differences of the predicted values and observed values (**r**esidual sum of squares, RSS), i.e. $\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

The R^2 is represented by these measures as follows:

$$R^2 = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS}$$

To penalize a model for the number of predictors (K) while considering the number of observations (N), the adjusted R^2 can also be used, particularly for model comparison:

$$\overline{R^2} = 1 - \frac{RSS/(N-K)}{TSS/(N-K)}$$

In our model, the values of R^2 and $\overline{R^2}$ show that our model fits data very well and the predictors describe large part of the variation of the response:

```
summary(mpgMod)$r.squared # R-squared
```

```
## [1] 0.8398903
```

```
summary(mpgMod)$adj.r.squared # Adjusted R-squared
```

```
## [1] 0.8227357
```

8.2 Functional form of the model is correct

The `pairs` plot seems to suggest a non-linear relationship between some variables. Thus, there is reason to expect that the transformation of observed values may result in a model with better fit. We can estimate the correctness of a model specification with a RESET test. This involves testing whether or not the coefficients of exponentiated predicted response values (e.g. $\hat{y}^2$) are zero or not when included in the initial model.

```
anova(mpgMod,
      lm(mpg ~ hp + wt + factor(am) + I(mpgMod$fitted^2) + I(mpgMod$fitted^3), mtcars))
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ hp + wt + factor(am)
## Model 2: mpg ~ hp + wt + factor(am) + I(mpgMod$fitted^2) + I(mpgMod$fitted^3)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     28 180.29
## 2     26 126.79  2    53.502 5.4857 0.01029 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

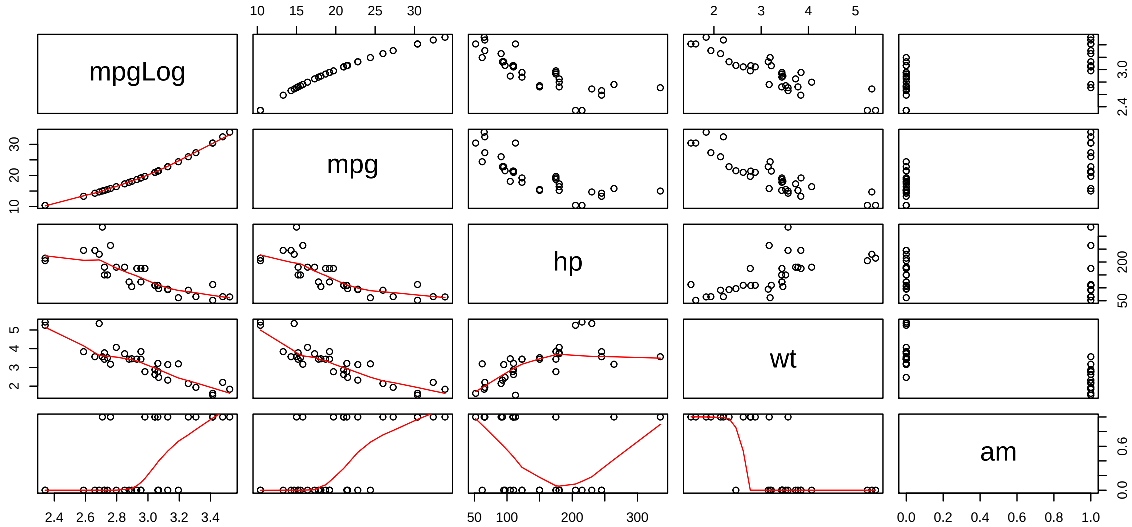
H0: Functional form of the model is correct
H1: Functional form of the model is not correct

A p value of F-test lower than a a critical value ($\alpha = 0.05$) suggests that the model is incorrectly specified.

8.2.1 Log-linear or log-log relationship

Model fit might be improved by using logged values of the response or predictor variables. For example, we can compare logged and not logged values of fuel consumption (`mpg`) in their relationship with other variables.

```
list(mpgLog = log(mtcars$mpg),
     mtcars[, c('mpg', 'hp', 'wt', 'am')]) %>%
  pairs(lower.panel = panel.smooth)
```



Logged fuel consumption (`mpg`) does seem to be more linearly related to weight (`wt`) than not logged values, so we can try to estimate a model with logged response variable.

```
lm(I(log(mpg)) ~ hp + wt + factor(am), mtcars) %>% summary
```

```
##
## Call:
## lm(formula = I(log(mpg)) ~ hp + wt + factor(am), data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.17137 -0.06955 -0.03865  0.07218  0.26567
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.7491397  0.1165798  32.159  < 2e-16 ***
## hp          -0.0016850  0.0004237  -3.976 0.000448 ***
## wt          -0.1757558  0.0399224  -4.402 0.000142 ***
## factor(am)1  0.0516749  0.0607202   0.851 0.401970
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1119 on 28 degrees of freedom
## Multiple R-squared:  0.8724, Adjusted R-squared:  0.8587
## F-statistic: 63.79 on 3 and 28 DF,  p-value: 1.24e-12
```

Here we can't use `anova` to test or R^2 to estimate the improvement of model specification as the variances of response variables are different. It's important to note that in log-linear and log-log models the coefficients represent elasticities, i.e. not absolute values but *per cent* changes in values of response (log-linear) or both, predictors and response (log-log).

8.2.2 Polynomials

Another way to obtain a better model fit is by using polynomials. Commonly, exponentiated values of predictors are used. For example, we can add the exponents of 2 of horsepower (`hp`) and weight (`wt`) to the model and test the result.

```
anova(mpgMod,
      lm(mpg ~ hp + I(hp^2) + wt + I(wt^2) + factor(am), mtcars))
```

```
## Analysis of Variance Table
```

```
##
## Model 1: mpg ~ hp + wt + factor(am)
## Model 2: mpg ~ hp + I(hp^2) + wt + I(wt^2) + factor(am)
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1     28 180.29
## 2     26 122.86  2    57.436 6.0776 0.00683 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

H0: Model with polynomials does not have lower residual sum of squares
H1: Model with polynomials does have lower residual sum of squares

Here, a statistically significant difference at $\alpha = 0.05$ suggests a better fit with exponentiated values.

8.3 Model parameters are statistically significant

Determining that the parameters in the model are statistically significant allows us to interpret the values of coefficients and to a certain extent makes sure that they represent more than just random noise in the data.

8.3.1 Individual significance of parameters (t-test)

To test the significane of the parameters separately we can use t-test. For each coefficient, we can calculate the value of the t-statistic using estimated value and corresponding standard errors and then estimate the probability of the aquired t value. The `lm` function does all this for us:

```
summary(mpgMod)$coefficients
```

```
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept) 34.00287512 2.642659337 12.866916 2.824030e-13
## hp          -0.03747873 0.009605422 -3.901830 5.464023e-04
## wt          -2.87857541 0.904970538 -3.180850 3.574031e-03
## factor(am)1  2.08371013 1.376420152  1.513862 1.412682e-01
```

H0: The coefficient is zero, i.e. $\beta = 0$
H1: Thecoefficient is not zero, i.e. $\beta \neq 0$

In our case horsepower and weight have a statistically significant effect on fuel consumption while transmisson does not ($\alpha = 0.05$). It's worth noting that a statistically insignificant parameter should not be excluded from a model when there is a valid causal relationship from a theoretical point of view.

8.3.2 Combined significance of parameters (f-test)

F-test can be used to compare models by comparing residual sums of squares. First, we can test if the model with parameters (full model) fits data better than a model with only the intercept (reduced model). Although this is also reported by `summary.lm`, we can use analysis of variance to test this explicitly:

```
anova(lm(mpg ~ hp + wt + factor(am), mtcars), # Full model
      lm(mpg ~ 1, mtcars)) # Reduced model
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ hp + wt + factor(am)
## Model 2: mpg ~ 1
##   Res.Df     RSS Df Sum of Sq     F    Pr(>F)
## 1     28  180.29
## 2     31 1126.05 -3   -945.76 48.96 2.908e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

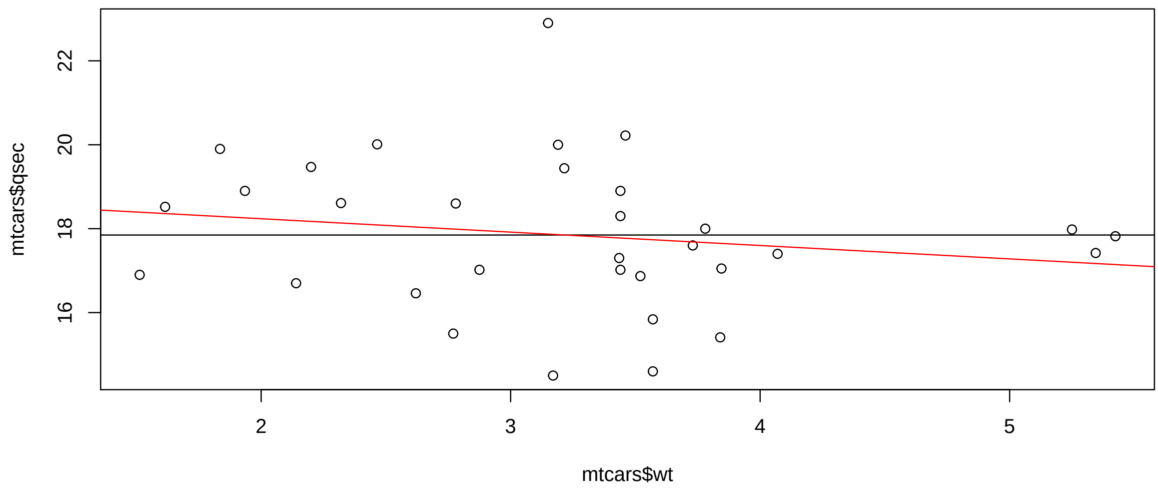
H0: All coefficients are simultaineously zero
H1: At least one of the coefficients is not zero 0

In our model at least one of the coefficients is not zero. But we could, for example, also estimate the effect of weight of a car (`wt`) on it's 1/4 mile time (`qsec`). In addition to an F-test, we can also visually confirm that the slope is not very different from zero.

```
# F-test
redMod <- lm(qsec ~ 1, mtcars) # Reduced model
fullMod <- lm(qsec ~ wt, mtcars) # Full model
anova(fullMod, redMod)
```

```
## Analysis of Variance Table
##
## Model 1: qsec ~ wt
## Model 2: qsec ~ 1
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     30 95.966
## 2     31 98.988 -1   -3.0217 0.9446 0.3389
```

```
# Visual
plot(mtcars$wt, mtcars$qsec)
abline(redMod) # Reduced model
abline(fullMod, col = 'red') # Full model
```



A second use for the F-test is to compare nested models. We can test if the coefficients for horsepower (`hp`) and transmission (`am`) are **simultaineously** significantly different from zero.

```
anova(lm(mpg ~ hp + wt + factor(am), mtcars),
      lm(mpg ~ wt, mtcars))
```

```
## Analysis of Variance Table
##
## Model 1: mpg ~ hp + wt + factor(am)
## Model 2: mpg ~ wt
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     28 180.29
## 2     30 278.32 -2   -98.031 7.6123 0.002291 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

H0: All coefficients in full but not in reduced model are zero
H1: At least one of the coefficients in full but not in reduced model are not zero

Insignificant F-statistic shows that we can reject the null hypothesis that the coefficients of `hp` and `factor(am)` are simultaineously zero.

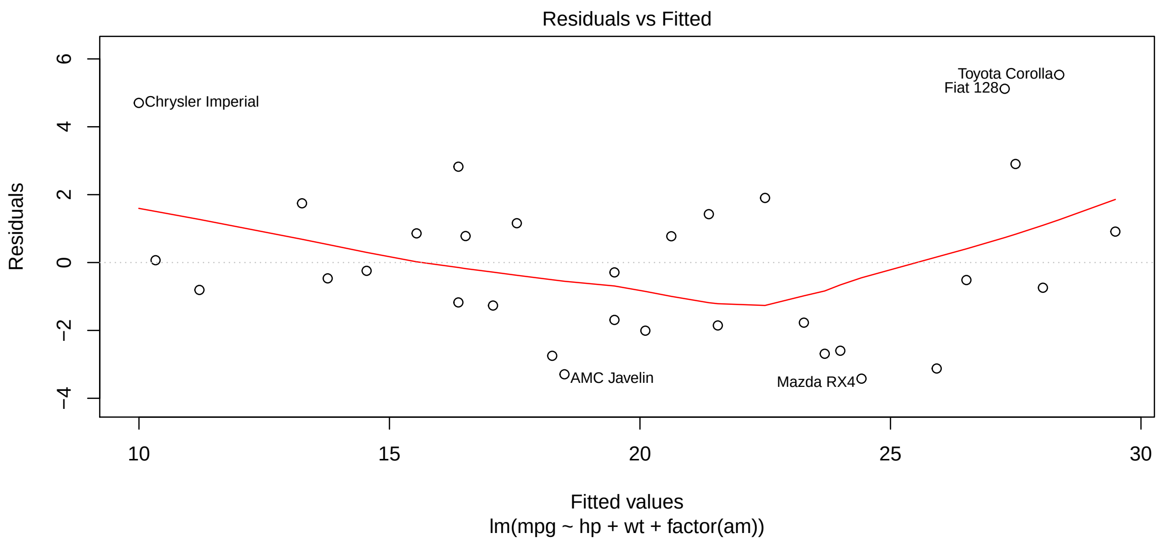
8.4 Model is not sensitive to individual observations

Even a small number of extreme observations can substantially alter model parameters in least squares estimation. Thus, it's important to be aware of atypical observatons and deal with them.

8.4.1 Residuals

A simple approach is to take a look at residuals at different fitted values. Residuals can be visualized by calling the 1st plot of `plot.lm` function. The function accepts `labels.id` and `id.n` as argumets which respectively set the vector used for point labels and the number of labels to add (starting from extreme values). This allows to understand which observations represent outliers in modelled relationship.

```
plot(mpgMod, which = 1, labels.id = rownames(mtcars), id.n = 5) # Residuals vs fitted
```

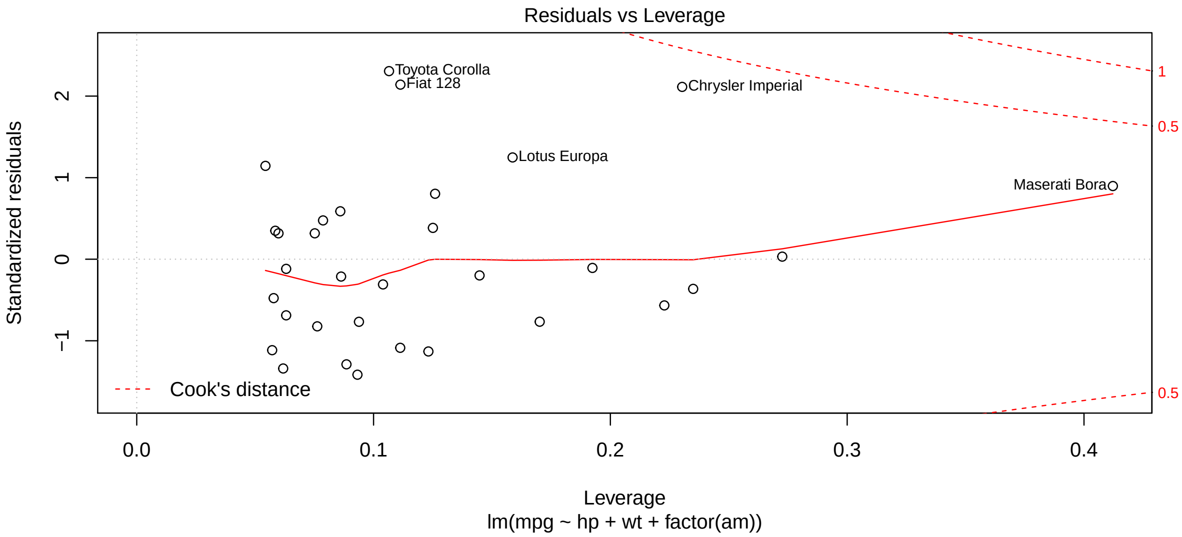


The observations with residuals that markedly diverge from 0 are outliers and may have a large influence on model parameters.

8.4.2 Leverage of observations

A more reliable method for detecting influential observations is to calculate leverage for each observation. Because predicted responses $\hat{y}$ are equal to matrix $H = X(X^TX)^{-1}X^T$ multiplied by observed responses y, the value of $h_{ii} = [H]_{ii}$ expresses the leverage of ith observation. That is, the leverage score h_{ii} represents the weight of ith observaton on predicted response $\hat{y}_i$. Leverages (along with residuals) can be visualized by calling the 5th plot of `plot.lm`.

```
plot(mpgMod, which = 5, id.n = 5) # Residuals vs leverage
```



The value of leverage score h_{ii} is between 0 and 1, thus a leverage over 0.4 (Maserati Bora) is rather high.

8.4.3 Influence of observations (Cook's distance)

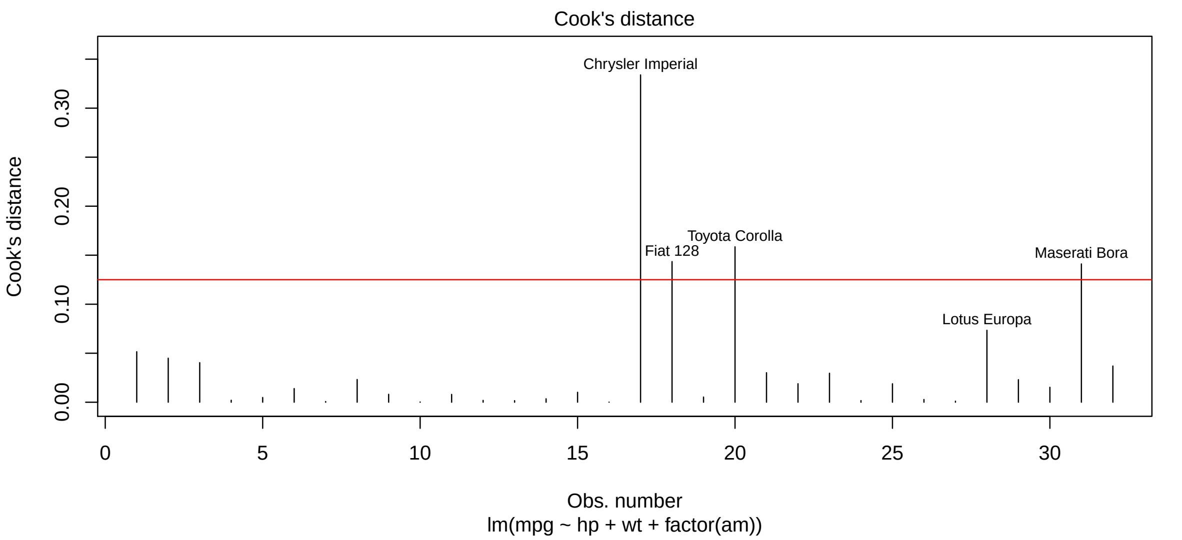
Influence of obsevations can also be determined by calculating how much model parameters change if an observation is omitted. This is what Cook's distance estimate D_i for observation i represents:

$$D_i = \frac{\sum_{j=i}^{n} (\hat{y}_j - \hat{y}_{j(i)})^2}{ps^2}$$

,

which, put plainly, is the sum of all differences in $\hat{y}$ when observation i is omitted, divided by number of parameters p multiplied by mean squared error s^2. Cook's distance can be plotted by calling 4th plot of `plot.lm`. A Cook's distance value that is higher than 4 divided by number of observations may be considered as influential, although there are less conservative suggestions for tresholds.

```
plot(mpgMod, which = 4, id.n = 5) # Cook's distance
abline(h = 4/nrow(mtcars), col = 'red')
```



When following the suggestion of 4/N for treshold, there are 4 influential observations in our model.

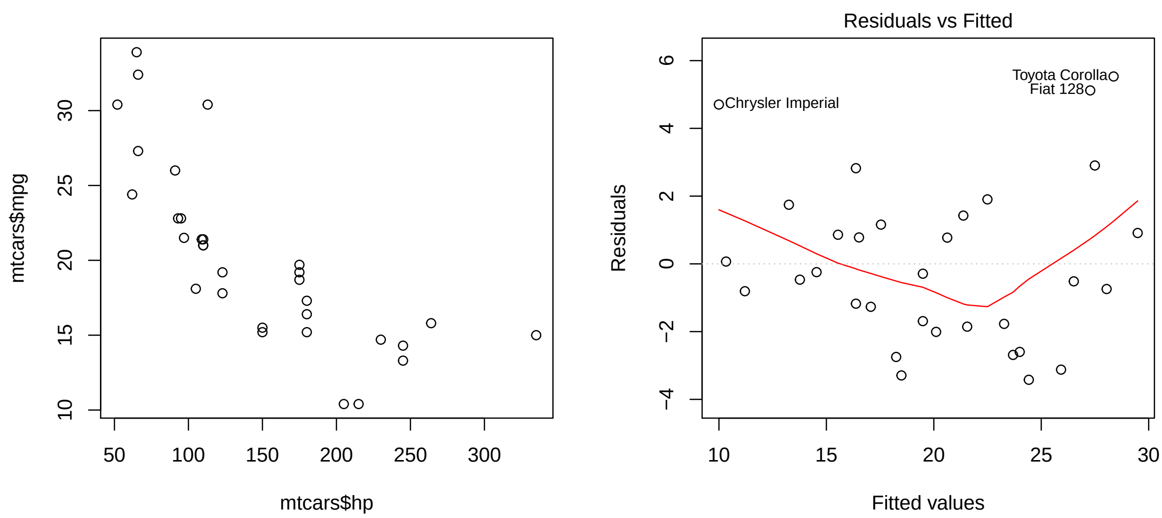
8.5 Error term is independent and has constant variance

Serial correlation and non-constant variance of residuals imply that model standard errors and thus respective p-values for coefficients are incorrect. In order to interpret the model parameters it's important to make sure that homoscedasticity nor autocorrelation are present and use corrected standard errors if otherwise.

8.5.1 Heteroscedasticity

Heteroscedasticity occurs when variance of error term for each observation might be different, i.e. is not constant, i.e. $var(\varepsilon|X) \neq \sigma^2$. Heteroscedasticity is usually evident when variables are plotted against each other or when residuals are plotted at different values of response variables.

```
par(mfrow = 1:2)
plot(mtcars$hp, mtcars$mpg)
plot(mpgMod, which = 1)
```




```
dev.off()
```

```
## null device
##           1
```

We can notice the lower variance of fuel consumption (`mpg`) at higher values of horsepower (`hp`).

One way to assess heteroscedasticity is to test whether variance of residuals is dependent on values of the predictors. This procedure is called Breusch-Pagan test.

```
modBp <- lm(mpgMod$residuals^2 ~ hp + wt + factor(am), mtcars)
bpTest <- nobs(modBp) * summary(modBp)$r.squared # Test statistic
pchisq(bpTest, df = modBp$rank - 1, lower.tail = FALSE)
```

```
## [1] 0.1366163
```

An alternative is to use the `bptest` function from `lmtest` package.

```
lmtest::bptest(mpgMod)
```

```
##
##   studentized Breusch-Pagan test
##
```

```
## data:  mpgMod
## BP = 5.534, df = 3, p-value = 0.1366
```

H0: Homoscedasticity
H1: Heteroscedasticity

A χ^2-test indicates that variance of residuals is independent of values of the predictors ($\alpha = 0.05$) and we can assume homoscedasticity.

8.5.2 Autocorrelation

Autocorrelation occurs when error terms of different observations are correlated with each other, i.e. $cov(\varepsilon_i \varepsilon_j | X) \neq 0, i \neq j$. Autocorrelatoin can be tested with Breusch-Godfrey test (`bgtest`) from `lmtest` package.

```
lmtest::bgtest(mpgMod)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  mpgMod
## LM test = 2.156, df = 1, p-value = 0.142
```

H0: No autocorrelation
H1: Autocorrelation

A p-value of 0.142 indicates lack of autocorrelation in our model ($\alpha = 0.05$).

8.5.3 Robust standard errors

Suppose that we have a model where heteroscedasticity is present (`wt ~ hp`, adding horsepower makes cars heavier).

```
wtMod <- lm(wt ~ hp, mtcars)
summary(wtMod)
```

```
##
## Call:
## lm(formula = wt ~ hp, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.41757 -0.53122 -0.02038  0.42536  1.56455
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.838247   0.316520   5.808 2.39e-06 ***
## hp          0.009401   0.001960   4.796 4.15e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7483 on 30 degrees of freedom
## Multiple R-squared:  0.4339, Adjusted R-squared:  0.4151
## F-statistic:    23 on 1 and 30 DF,  p-value: 4.146e-05
```

```
lmtest::bptest(wtMod)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  wtMod
## BP = 7.6716, df = 1, p-value = 0.00561
```

We can expect that these standard errors are not reliable, so they need to be corrected. This involves calculating heteroscedasticity consistent (HC) standard errors. This can be done by plugging a defined symmetric diagonal matrix $\Omega = diag(\omega_1, ..., \omega_i)$ into coefficient covariance matrix and calculating standard errors from the result (Zeileis 2004[1]). There are different estimators for ωi but in most cases we can use ε_i^2. Such covariance matrix with can be calculated with `vcovHC` function from `sandwhich` package setting `HC0` as type. We can get the t- and p-values with HC standard errors with `coeftest` function from `lmtest` pacakge by inserting the new covariance matrix into initial model.

```
wtModVcov <- sandwich::vcovHC(wtMod, 'HC0') # Calculate covariance matrix
wtModVcov %>% diag %>% sqrt # Get robust standard errors
```

```
## (Intercept)          hp
## 0.339692165 0.002611827
```

```
lmtest::coeftest(wtMod, vcov = wtModVcov) # Find p-values
```

```
##
## t test of coefficients:
##
##              Estimate Std. Error t value  Pr(>|t|)
## (Intercept) 1.8382467  0.3396922  5.4115 7.288e-06 ***
## hp          0.0094010  0.0026118  3.5994  0.001133 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see that while the p-value for `hp` coefficient calculated with robust standard errors is still statistically significant ($\alpha = 0.05$), it's much higher than before.

8.6 Error term is uncorrelated to predictor(s)

When error term is correlated to a predictor, i.e. $cov(x_i, \varepsilon) \neq 0$, the model parameters are biased and not consistent. The predictors causing this are endogenous to the model and in such cases we need to find an instrumental variable (instrument) that is correlated to the endogenous predictors but not the error term.

Let's estimate the effect of engine size (`disp`, displacement) and horsepower (`hp`) on fuel consumption (`mpg`).

```
olsMod <- lm(mpg ~ hp + disp, mtcars)
summary(olsMod)
```

```
##
## Call:
## lm(formula = mpg ~ hp + disp, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7945 -2.3036 -0.8246  1.8582  6.9363
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.735904   1.331566  23.083  < 2e-16 ***
## hp          -0.024840   0.013385  -1.856 0.073679 .
## disp        -0.030346   0.007405  -4.098 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.127 on 29 degrees of freedom
```

[1]Zeileis, A. (2004). Econometric Computing with HC and HAC Covariance Matrix Estimators. Journal of Statistical Software, 11(1), 1–17. https://doi.org/10.18637/jss.v011.i10

```
## Multiple R-squared:  0.7482, Adjusted R-squared:  0.7309
## F-statistic: 43.09 on 2 and 29 DF,  p-value: 2.062e-09
```

There's a statistically significant relationship. However, theoretically more weight (`wt`) and number of carburetors (`carb`) require a larger engine (`disp`), so it could actually be these two variables that increase fuel consumption. We can estimate the instrumental variable regression by two-stage least squares (2SLS) either manually or use the `ivreg` function from `AER` package. In the second stage there's an option to either replace the endogenous variable with predicted values from 1st stage or just add residuals from the 1st stage. Note that the standard errors in the 2nd stage are incorrect when calculated manually as below.

```
# Manually
ivModS1 <- lm(disp ~ hp + wt + carb, mtcars)
(ivModS2Fit <- lm(mpg ~ hp + ivModS1$fitted, mtcars)) # Replace disp
```

```
##
## Call:
## lm(formula = mpg ~ hp + ivModS1$fitted, data = mtcars)
##
## Coefficients:
##    (Intercept)              hp  ivModS1$fitted
##       30.88810        -0.01447        -0.03760
```

```
(ivModS2Res <- lm(mpg ~ hp + disp + ivModS1$residuals, mtcars)) # Add residuals
```

```
##
## Call:
## lm(formula = mpg ~ hp + disp + ivModS1$residuals, data = mtcars)
##
## Coefficients:
##       (Intercept)                 hp               disp
##          30.88810           -0.01447           -0.03760
## ivModS1$residuals
##           0.03368
```

```
# AER::ivreg
ivModS2Ivreg <- AER::ivreg(mpg ~ hp + disp | hp + wt + carb, data = mtcars)
summary(ivModS2Ivreg, vcov = sandwich::sandwich, diagnostics = T)
```

```
##
## Call:
## AER::ivreg(formula = mpg ~ hp + disp | hp + wt + carb, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0066 -2.3808 -0.3478  1.8353  6.6258
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 30.888099   1.367368  22.589  < 2e-16 ***
## hp          -0.014474   0.009231  -1.568    0.128
## disp        -0.037596   0.007120  -5.280 1.16e-05 ***
##
## Diagnostic tests:
##                  df1 df2 statistic  p-value
## Weak instruments   2  28    66.105 2.48e-11 ***
## Wu-Hausman         1  28     5.122   0.0316 *
## Sargan             1  NA     6.028   0.0141 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.178 on 29 degrees of freedom
## Multiple R-Squared: 0.7399,  Adjusted R-squared: 0.722
## Wald test:  52.6 on 2 and 29 DF,  p-value: 2.252e-10
```

The `summary` output of `ivreg` includes three diagnostic tests given that `diagnostics = T` is passed to the function.

8.6.1 Weak instruments

The weakness of instruments can be estimated with an F-test to determine weather the instrument has an effect on the endogenous variable.

```
anova(lm(disp ~ hp, mtcars), # Model without instruments
      lm(disp ~ hp + wt + carb, mtcars)) # Model with instruments
```

```
## Analysis of Variance Table
##
## Model 1: disp ~ hp
## Model 2: disp ~ hp + wt + carb
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     30 178284
## 2     28  38378  2    139906 51.037 4.587e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

H0: All instruments have no effect
H1: At least one of the instruments has an effect

8.6.2 Wu-Hausmann test

The coefficient for `disp` is efficient but potentially inconsistent. We can assess the consistency of the predictors in the OLS model by testing whether the model parameters given by OLS and IV models are different. This can be done with the (Durbin-)Wu-Hausman test where the test statistic is as follows.

$$H = (b_{IV} - b_{OLS})^T (var(b_{IV}) - var(b_{OLS}))^+ (b_{IV} - b_{OLS})$$

```
(coefDiff <- coef(ivModS2Ivreg) - coef(olsMod)) # Difference in coefficient vectors
```

```
##  (Intercept)           hp         disp
##  0.152194751  0.010365781 -0.007249964
```

```
(covDiff <- vcov(ivModS2Ivreg) - vcov(olsMod)) # Difference of covariance matrices of coefficients
```

```
##               (Intercept)            hp          disp
## (Intercept)  0.0654566494  1.955037e-04 -3.642333e-04
## hp           0.0001955037  3.768635e-05 -2.480745e-05
## disp        -0.0003642333 -2.480745e-05  1.735066e-05
```

```
(whStat <- as.vector(t(coefDiff) %*% solve(covDiff) %*% coefDiff)) # Wu-Hausman statistic
```

```
## [1] 3.029394
```

```
pchisq(whStat, df = olsMod$rank - 1, lower.tail = F) # Find the significance of the test statistic
```

```
## [1] 0.2198747
```

H0: The predictor in the OLS model is not correlated to the error term and consistent
H1: The predictor in the OLS model is correlated to the error term and not consistent

!!! Different results !!!

8.6.3 Sargan test

When we use more than one instrument, the restrictions may be overidentified. We can test this by calculationg a test statistic from the effects of exogenous variables on the residuals from the 2nd stage of IV model.

```
sarMod <- lm(ivModS2Ivreg$resid ~ hp + wt + carb, mtcars)
sarStat <- summary(sarMod)$r.squared * nobs(sarMod)
sarDf <- 2 - 1 # Degrees of freedom: number of instruments - number of endogenous variables
1 - pchisq(sarStat, sarDf)
```

```
## [1] 0.01407694
```

H0: All instruments are valid
H1: At least one of the instruments is not valid

Here we reject the null hypothesis ($\alpha = 0.05$) and assume that either `wt` or `carb` is not a valid instrument.

8.7 Predictors can not be linearly predicted from others (no multicollinearity)

When one predictor variable can be linearly predicted from others, the model parameters are sensitive to changes in model or data. We can evaluate multicollinearity by calculating a variance inflation factor (VIF) for each predictor. VIF expresses the variaton of a predictor that can be explained by other predictors in the model.

$$VIF_i = \frac{1}{1 - R_i^2}$$

Hence, it's simple to caluculate manulally but we can also use the `vif` function from package `car`.

```
# Manually
for (i in attributes(mpgMod$terms)$term.labels) {
  formula <- paste(i, "~",
                   paste(setdiff(attributes(mpgMod$terms)$term.labels, i), collapse = "+"))
  model <- lm(formula, mtcars)
  print(1/ (1 - summary(model)$r.squared))
}
```

```
## [1] 2.088124
## [1] 3.774838
## [1] NA
```

```
# car::vif
car::vif(mpgMod)
```

```
##        hp        wt factor(am)
##  2.088124  3.774838  2.271082
```

The manual calculation did not yield a VIF value for `factor(am)` which should not be a problem since VIF is not very meaningful for nominal variables. A VIF value of >10 is usually considered as a sign of high multicollinearity.

8.8 TODO

- Explain log-linear etc. models

Chapter 9

Generalized linear models

9.1 Link functions in R

9.2 Binary

Logit, probit, ROC curve etc

Chapter 10

GLM for categorical variables

10.1 Ordered logit

10.2 Multinomial logit

10.3 Count data

Chapter 11

Panel data methods

11.1 Fixed and random effects

11.2 Panel data methods

Static, dynamic

Chapter 12

Impact evaluation

12.1 Experimental vs non-experimental data

Selection bias

12.2 Basic

Before-after, with-without, difference in differences

Matching

12.3 Instrumental variable

12.4 Regression discontinuity