Intro Data Analysis

Jan-Philipp Kolb and Alexander Murray-Watters

12 Juni 2018

Getting started

Preliminaries

- Usually big differences in knowledge and abilities of the participants - please tell, if it is too fast or slow.
- We have many **exercises** - at the end you can only learn on your own
- If there are questions - always ask
- R is more fun together - ask your neighbor

Outline

Day	Part	Topic
Monday	A1	Getting started
Monday	A2	How to get help
Monday	A3	GESIS Panel
Monday	A4	Data Import
Monday	A5	Data Export

Overview - why use R

Reasons for using R...

- ... because it is an**open source language**
- ... **Graphics**, **graphics**, **graphics**
- ... **it can be used in combination with other programs**
- ... relates to other languages, e.g. **data linking**
- ... **for automation**
- Vast Community - ... **to use the intelligence of other people ;-)**
- ...

Advantages of R

- R can be downloaded for **free**.
- R is a **scripting language**
- R is becoming more **popular**
- **Good** possibilities for **visualization**

R can be used in combination...

- **Calling Python from R**

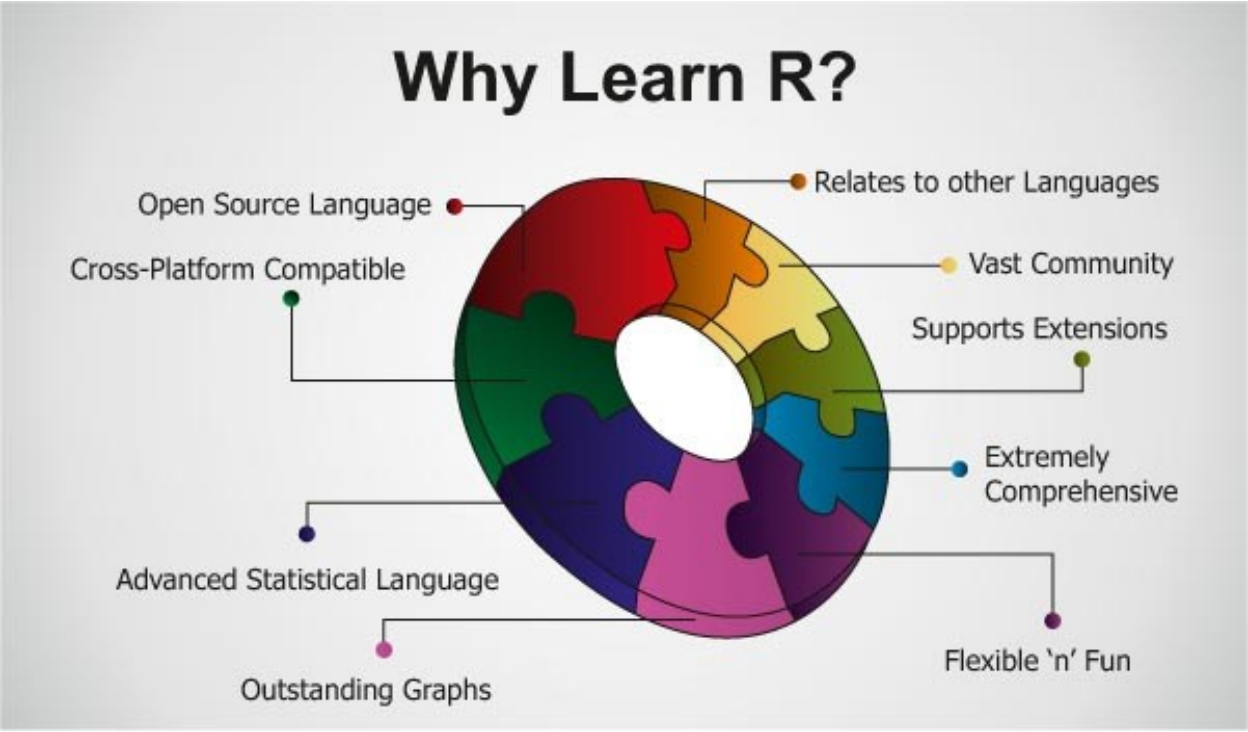


Figure 1: http://bigdatahadooppro.com/tag/advantages-of-using-r/

Figure 2:

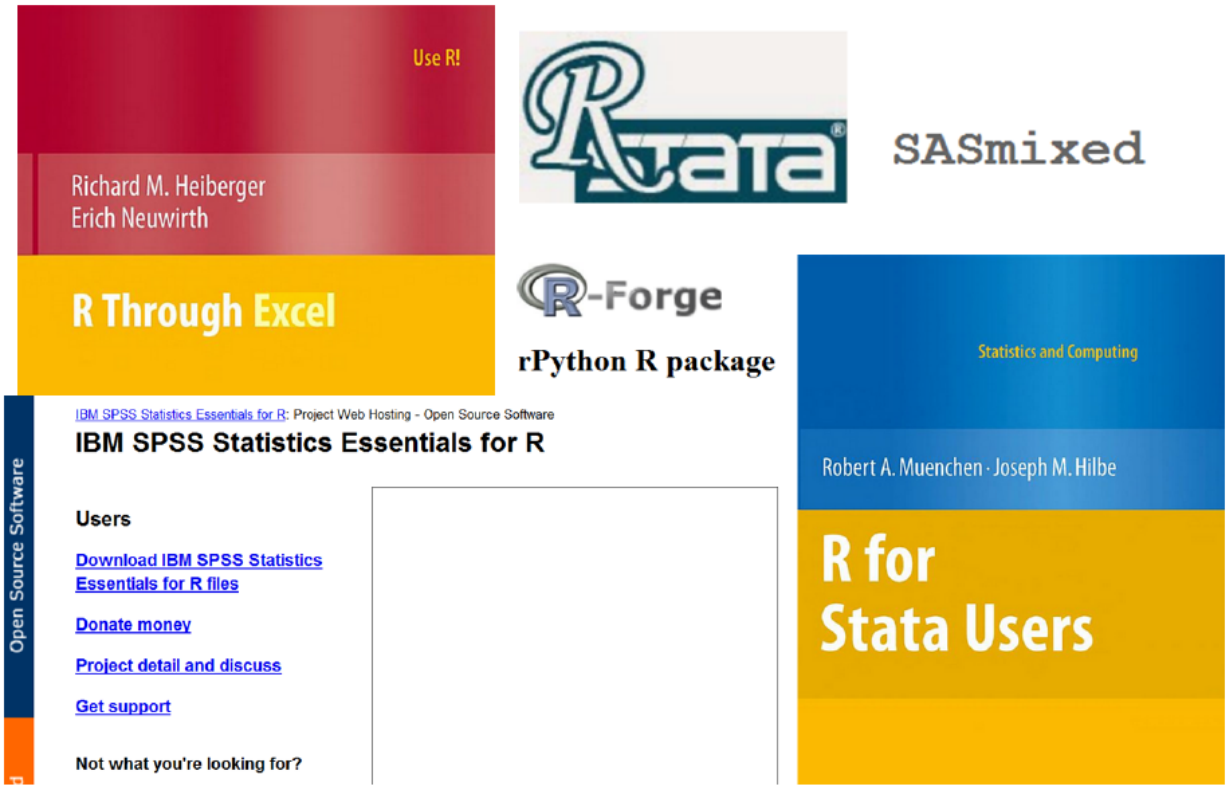


Figure 3: Interfaces to R**

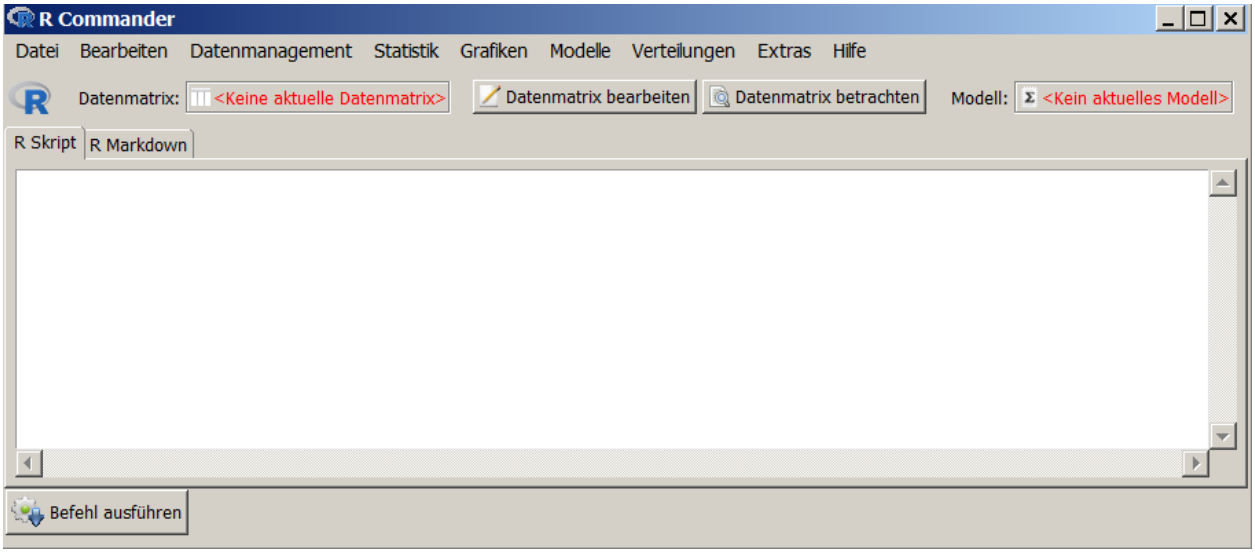


Figure 4: The Rcommander

- **R Through Excel** - A Spreadsheet Interface for Statistics, Data Analysis, and Graphics
- **Installing the Essentials for R for Statistics plug-in SPSS**
- **SASmixed: Data sets from "SAS System for Mixed Models"**
- **RStata: A Bit of Glue Between R and Stata**
- **Getting Started in R Stata Notes on Exploring Data**

R for SPSS user

Bob Muenchen - **R for SPSS and SAS Users**

- **R commander (Rcmdr)**

Use R because other programs have big bugs:

Problems with Excel

The popularity of R-packages

Where are the most active users?

-where is R activity the most concentrated

Download R:

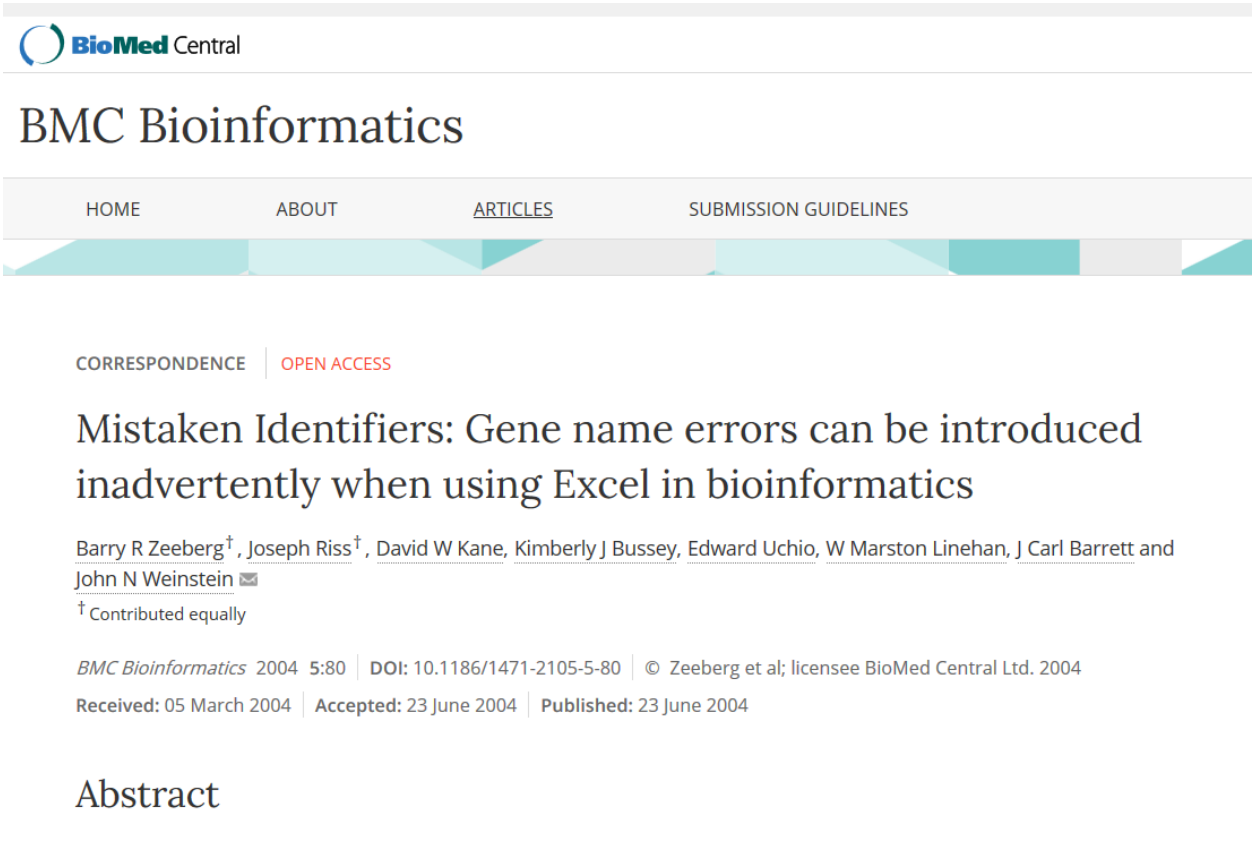
http://www.r-project.org/

FAQ: Reinhart, Rogoff, and the Excel Error That Changed History

By Peter Coy

PHOTOGRAPH BY GREGOR SCHUSTER

Figure 5:



BioMed Central

BMC Bioinformatics

HOME ABOUT ARTICLES SUBMISSION GUIDELINES

CORRESPONDENCE | OPEN ACCESS

Mistaken Identifiers: Gene name errors can be introduced inadvertently when using Excel in bioinformatics

Barry R Zeeberg†, Joseph Riss†, David W Kane, Kimberly J Bussey, Edward Uchio, W Marston Linehan, J Carl Barrett and John N Weinstein

† Contributed equally

BMC Bioinformatics 2004 **5**:80 | **DOI:** 10.1186/1471-2105-5-80 | © Zeeberg et al; licensee BioMed Central Ltd. 2004

Received: 05 March 2004 | **Accepted:** 23 June 2004 | **Published:** 23 June 2004

Abstract

Figure 6: Mistaken Identifiers in Excel

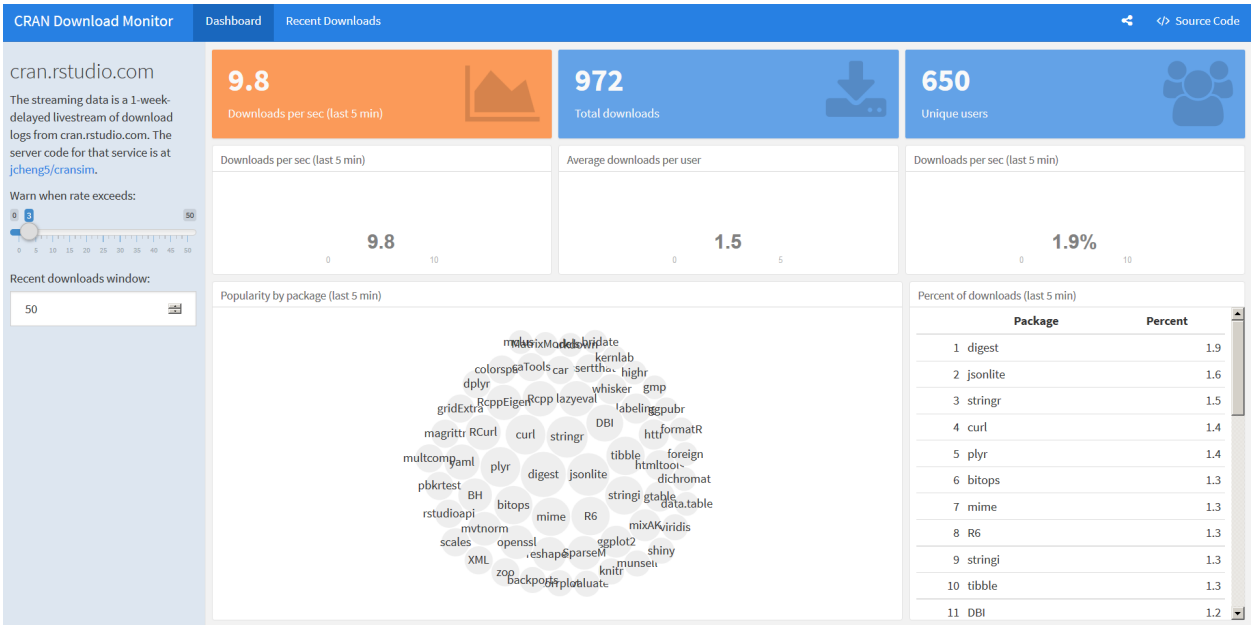


Figure 7: Downloads from CRAN

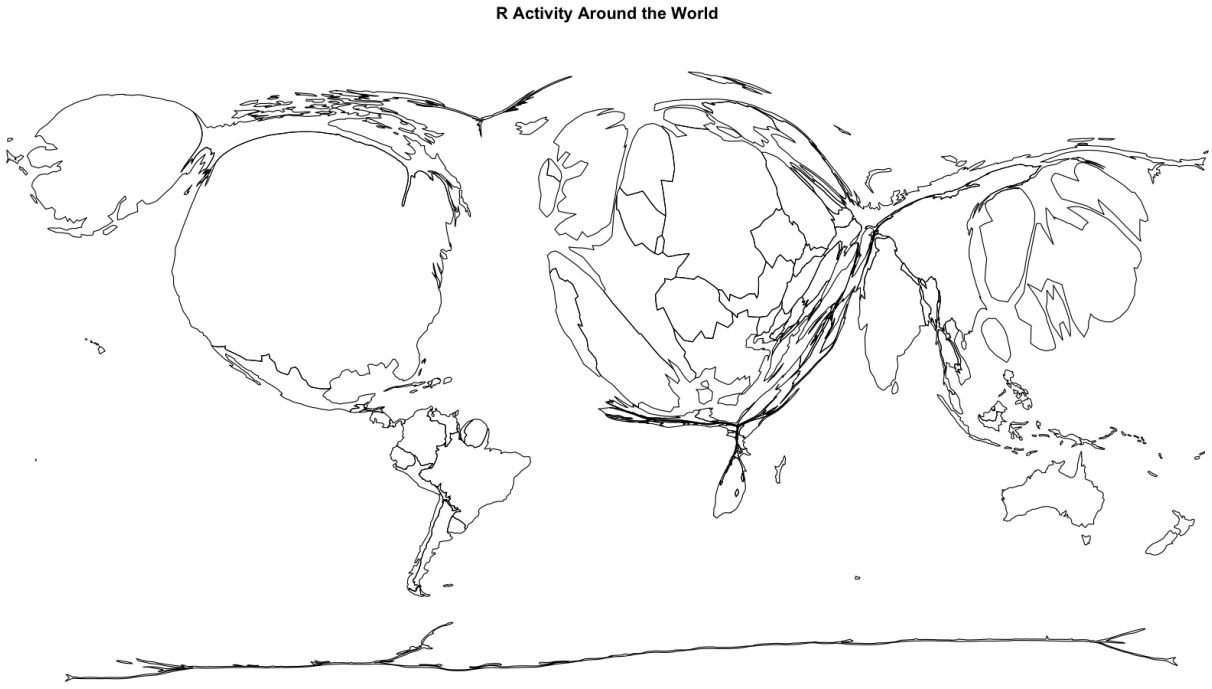


Figure 8: R activity around the world

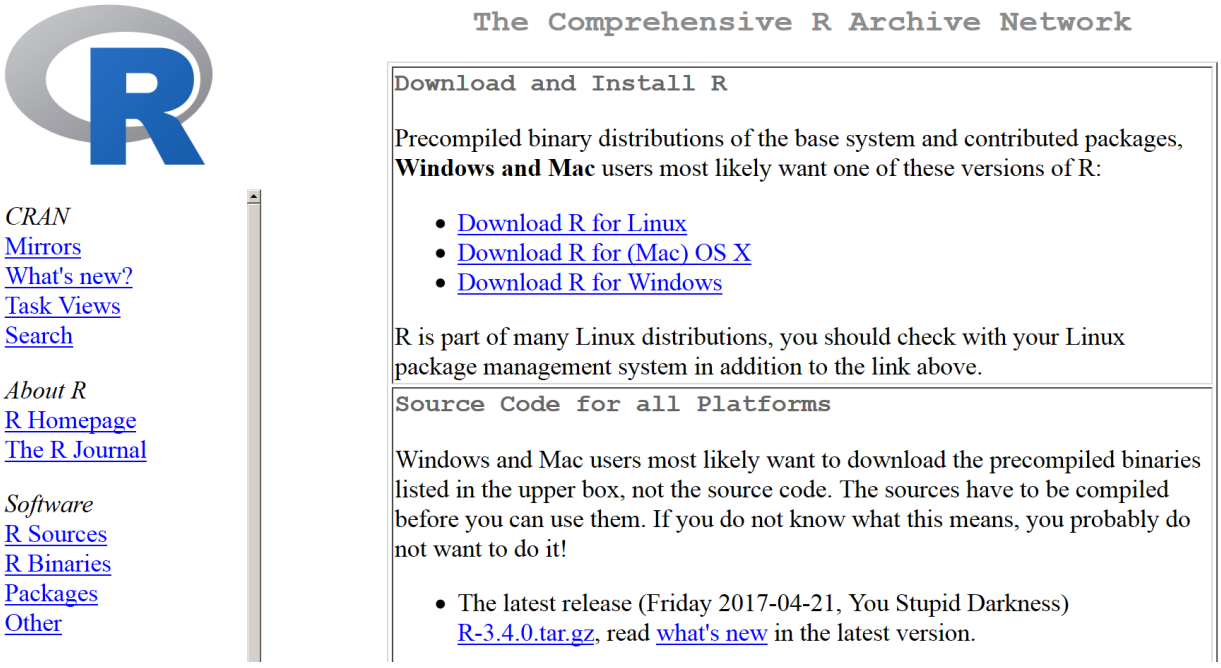


Figure 9: The CRAN website

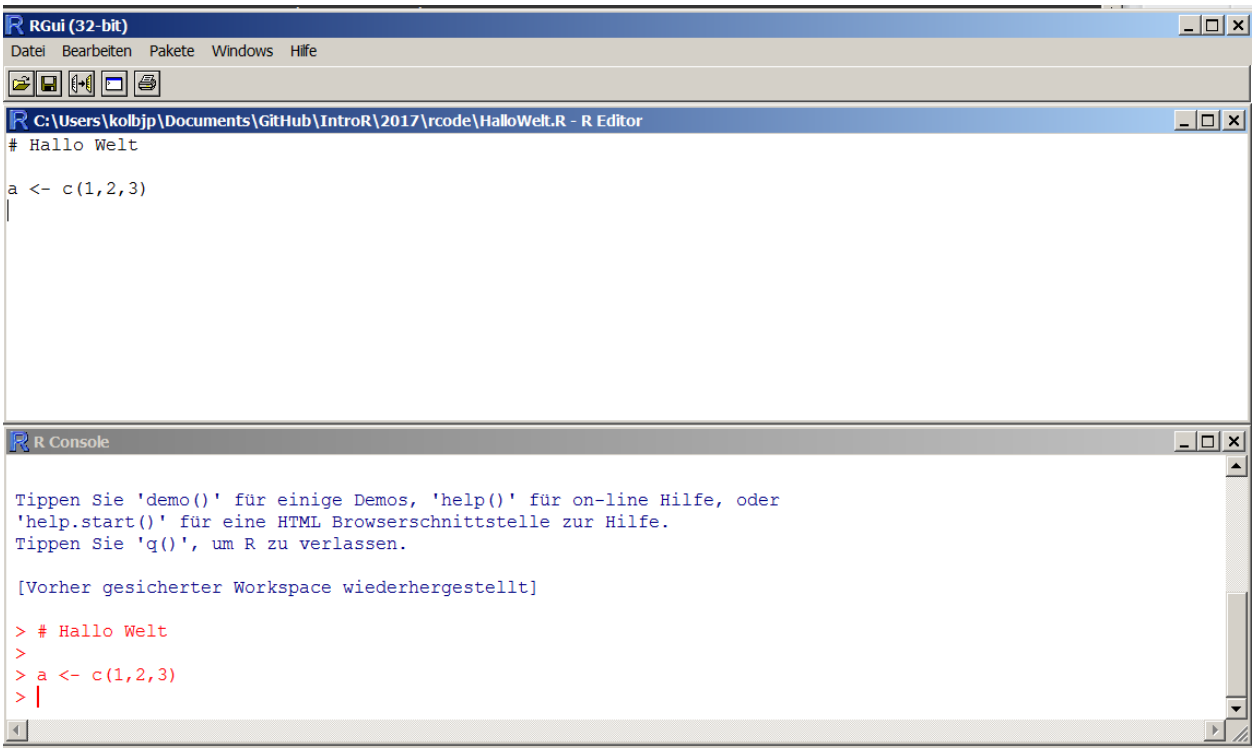


Figure 10:

Open Source Programm R

- R is a free, non-commercial implementation of the S programming language (developed by AT & T Bell Laboratories)
- Free participation - modular structure (growing number of packages)

This is base R:

Graphical user interface

But many people use a graphical user interface (GUI) or a integrated development interface (IDE).

For the following reasons:

- Syntax highlighting
- Auto-completion
- Better overview on graphics, libraries, files, ...

Various text editors / IDEs

- **Gedit** with R-specific Add-ons for Linux
- **Emacs** and ESS (Emacs speaks statistics)- An extensible, customizable, free/libre text editor — and more.
- I use **Rstudio!**

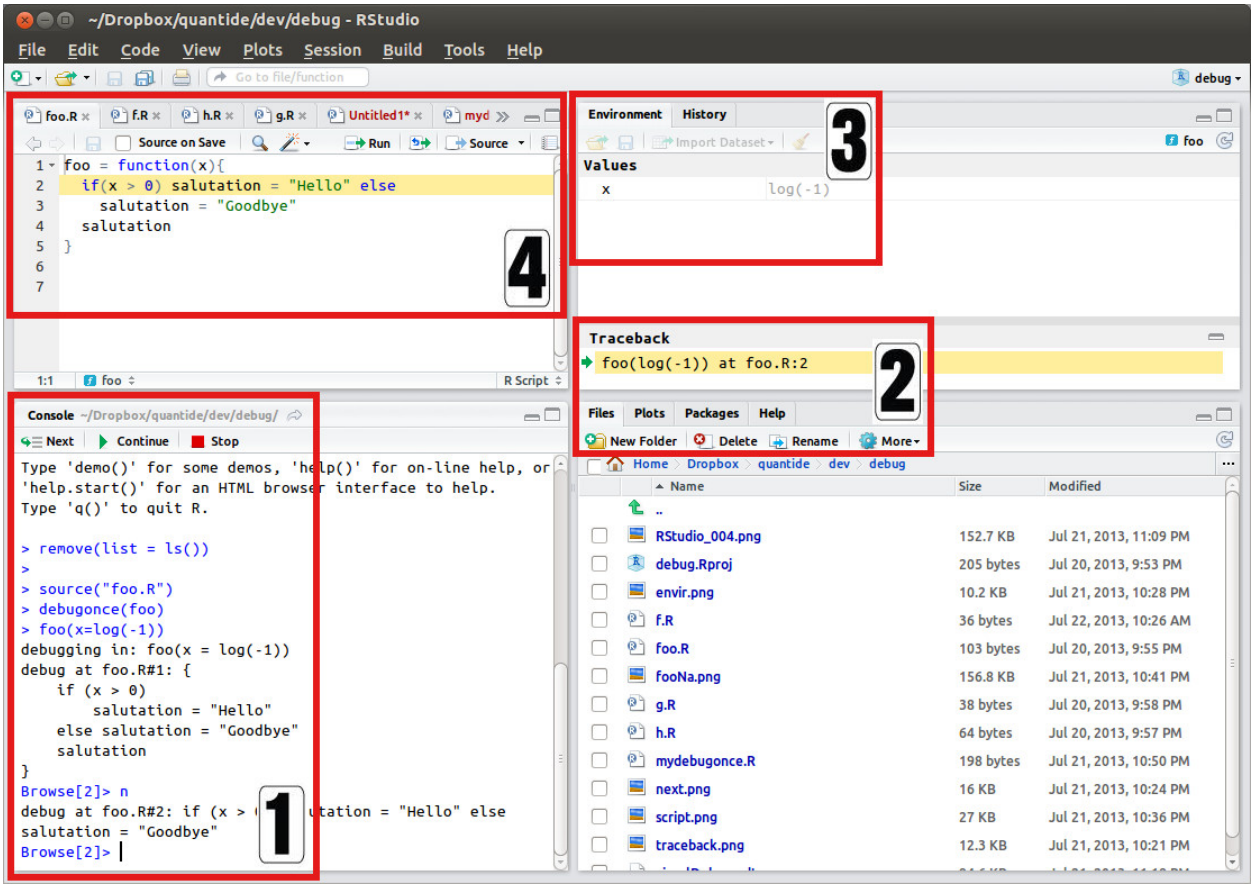


Figure 11: Overview Rstudio

Customizing RStudio

Customize Rstudio

- Six **reasons** to use Rstudio.
- RStudio Support - **Using the RStudio IDE**

http://www.rstudio.com/ide/docs/using/customizing

A1A Exercise - Preparation

- Check if R is installed on your computer.
- If not, download **R** and install it.
- Check if Rstudio is installed.
- If not - **install** Rstudio.
- Start RStudio. Go to the console (lower left window) and write

```
3+2
```

- If there is not already an editor open in the upper left window, then go to the file menu and open a new script. Check the date with `date()` and the R version with `sessionInfo()`.

```
date()
```

```
sessionInfo()
```

R is a object-orientiented language

Vectors and assignments

- R is a object-orientiented language
- `<-` is the assignment operator

```
b <- c(1,2) # create an object with the numbers 1 and 2
```

- A function can be applied to this object:

```
mean(b) # computes the mean
```

```
## [1] 1.5
```

With the following functions we can learn something about the properties of the object:

```
length(b) # b has the length 2
```

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

Object structure

```
str(b) # b is a numeric vector
```

```
##  num [1:2] 1 2
```

Functions in base-package

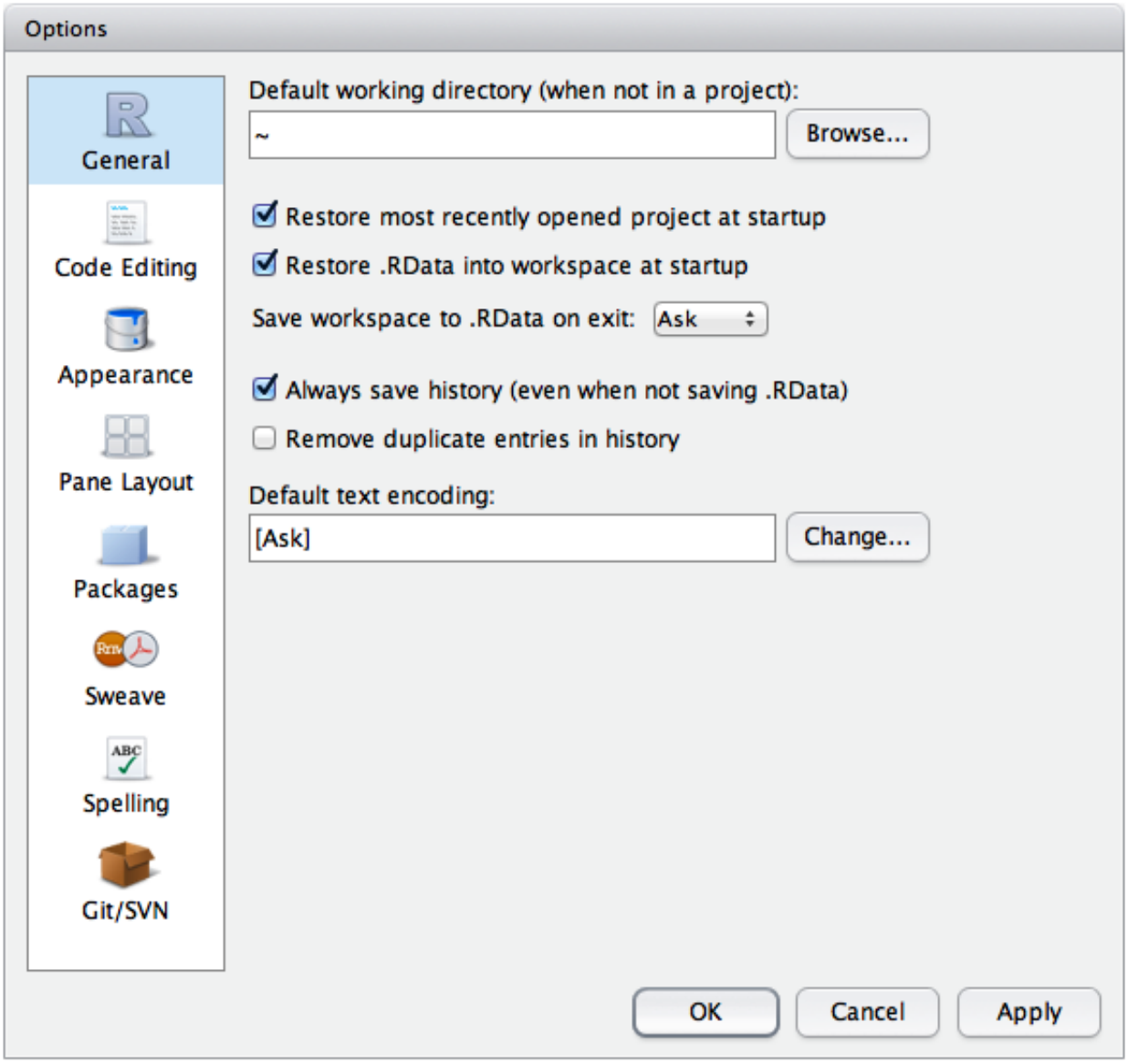

Options
General
Code Editing
Appearance
Pane Layout
Packages
Sweave
Spelling
Git/SVN
Default working directory (when not in a project):
~
Browse...
Restore most recently opened project at startup
Restore .RData into workspace at startup
Save workspace to .RData on exit:
Ask
Always save history (even when not saving .RData)
Remove duplicate entries in history
Default text encoding:
[Ask]
Change...
OK
Cancel
Apply

Figure 12:

Function	Meaning	Example
str()	Object structure	str(b)
max()	Maximum	max(b)
min()	Minimum	min(b)
sd()	Standard deviation	sd(b)
var()	Variance	var(b)
mean()	Mean	mean(b)
median()	Median	median(b)

These functions only need one argument.

Functions with more arguments

Other functions need more arguments:

Argument	Meaning	Example
quantile()	90 % Quantile	quantile(b,.9)
sample()	Draw a sample	sample(b,1)

```
quantile(b,.9)
```

```
## 90%
## 1.9
```

```
sample(b,1)
```

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

Examples - Functions with more than one argument

```
max(b)
```

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

```
min(b)
```

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

```
sd(b)
```

```
## [1] 0.7071068
```

```
var(b)
```

```
## [1] 0.5
```

Functions with one argument

```
mean(b)
```

```
## [1] 1.5
```

An Introduction to R

Table of Contents

Figure 13: Overview commands

```
median(b)
```

```
## [1] 1.5
```

Overview commands

http://cran.r-project.org/doc/manuals/R-intro.html

A1B Exercise - assignments and functions

Create a vector `b` with the numbers from 1 to 5 and calculate ...

1. the mean
2. the variance
3. the standard deviation
4. the square root from the mean

Where to find routines

- Many functions are included in basic R
- Many specific functions are integrated in additional libraries

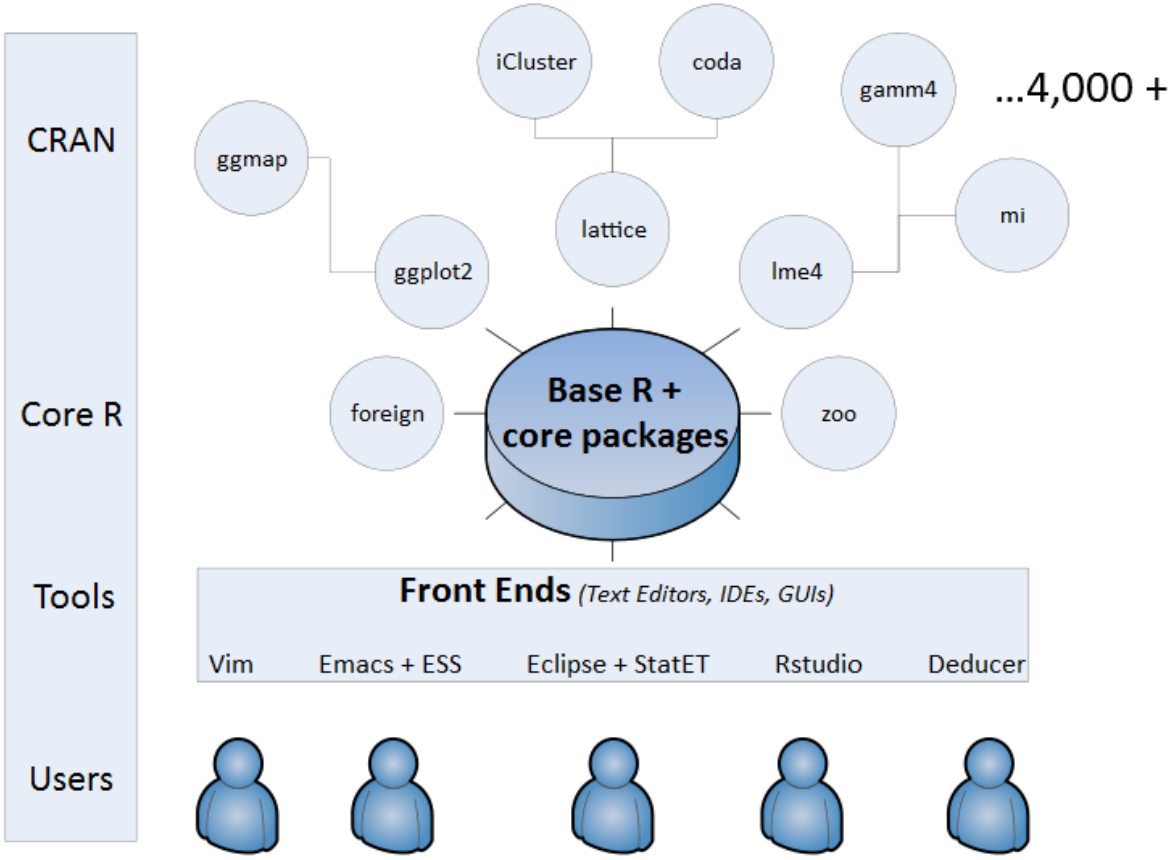


Figure 14: Overview R-packages

- R can be modularly extended by so-called packages or libraries
- The most important packages hosted on CRAN (12738 at Mo Jul 23)
- Further packages can be found e.g. at **bioconductor**

Overview R packages

Installation of packages

- The quotes around the package name are necessary for the command `install.packages`.
- They are optional for the command `library`.
- You can also use `require` instead of `library`.

```
install.packages("lme4")

library(lme4)
```

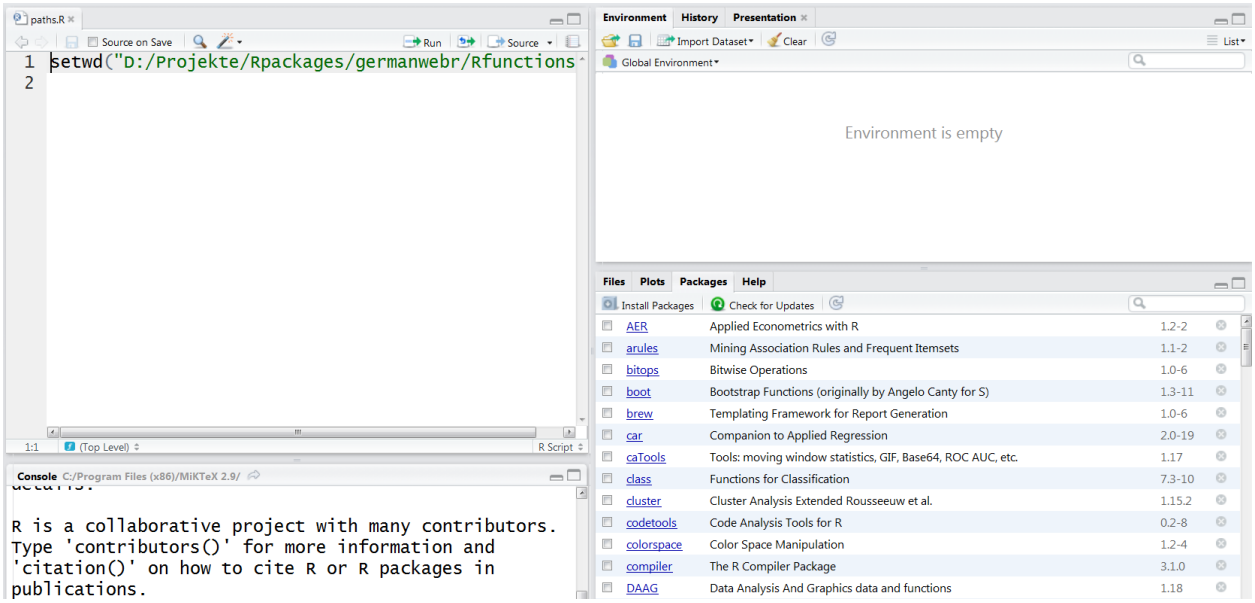


Figure 15: Package installation with Rstudio

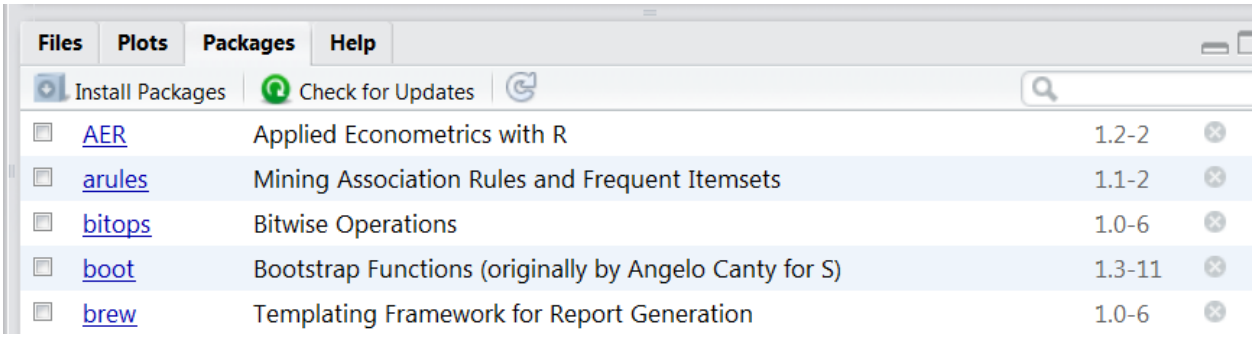


Figure 16: Existing packages

Installation of packages with RStudio

Existing packages and installation

Overview of many useful packages:

- Luhmann - **Table with many useful packages**

Other interesting packages:

- Package for Import/Export - `foreign`
- **`sampling`-package for survey Sampling**
- `xtable` Package for integrating LateX in R (**xtable Galerie**)
- **`dummies` package for creating dummies**
- **Package `mvtnorm` for getting a multivariate normal distribution**
- **Package `maptools` for creating maps**

Install packages from various sources

Install packages from CRAN Server

```
install.packages("lme4")
```

Install packages from Bioconductor Server

```
source("https://bioconductor.org/biocLite.R")
biocLite(c("GenomicFeatures", "AnnotationDbi"))
```

Install packages from Github

```
install.packages("devtools")
library(devtools)

install_github("hadley/ggplot2")
```

How do I get an overview

- **Discover packages recently uploaded to CRAN**
- Shiny web app that shows the **packages recently downloaded from CRAN**
- A **quick-list of useful packages**
- A list with the **best packages for data processing and analysis**
- **the 50 most used packages**

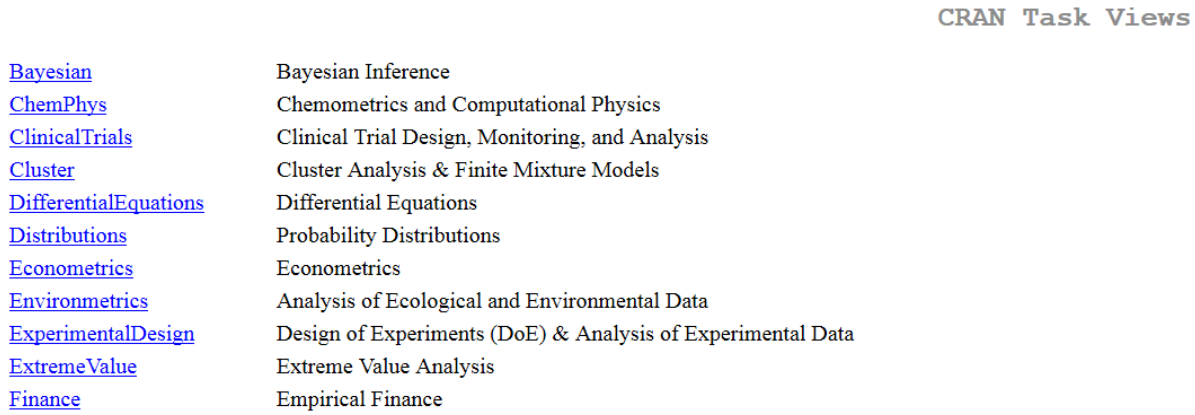


Figure 17:

CRAN Task Views

- For some topics all possibilities are arranged in R. (**Overview of Task Views**)
- Currently there are 35 task views.
- All packages of a task view can be installed with the following **command:**

```
install.packages("ctv")
library("ctv")
install.views("Bayesian")
```

A1C Exercise - additional packages

Go to https://cran.r-project.org/ and search for packages in the area where the packages are presented,...

- which are suitable for descriptive data analysis.
- to calculate regressions
- to read in external data records (e.g. SPSS data)
- to handle large amounts of data

Some links to read on

- **Why you should learn R first for data science**
- **RStudio – Infoworld 2015 Technology of the Year Award Recipient!**
- **Why the R programming language is good for business?**
- **Have a look at R-bloggers**
- **Comparisson between python and R**
- **R and Stata Side-by-side**
- **AWESOME R**
- **1000 R tutorials/Links**

How to get help

How to get help?

- **To get help in general:**

```
help.start()
```

- **Online documentation for most of the functions:**

```
help(name)
```

- Use `?` to get help.

```
?mean
```

- `example(lm)` gives an example for a linear regression

```
example(lm)
```

Vignettes

- Paper that present functions in the package
- Many reproducible examples
- But not every package has a vignette

```
browseVignettes()
```

- to get a vignette:

```
vignette("osmdata")
```

An example for a vignette - package `osmdata`

Demos

- for some packages you have demos:

```
demo() # shows all available demos
demo(package = "httr") # Show all demos in a package

# Run a specific demo:
demo("oauth1-twitter", package = "httr")
```

- if you run a demo, the code is shown in the console

```
demo(nlm)
```

The function `apropos`

- searches everything about the given string

```
apropos("lm")
```

https://cran.r-project.org/web/packages/osmdata/vignettes/osmdata.html

1. Introduction

`osmdata` is an R package for downloading and using data from OpenStreetMap (OSM). OSM is a global open access mapping project, which is free and open under the ODbL licence [@OpenStreetMap]. This has many benefits, ensuring transparent data provenance and ownership, enabling real-time evolution of the database and, by allowing anyone to contribute, encouraging democratic decision making and citizen science [@johnson_models_2017]. See the OSM wiki to find out how to contribute to the world's open geographical data commons.

Unlike the `OpenStreetMap` package, which facilitates the download of raster tiles, `osmdata` provides access to the vector data underlying OSM.

`osmdata` can be installed from CRAN with

```
install.packages("osmdata")
```

and then loaded in the usual way:

```
library(osmdata)
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. http://www.openstreetmap.org/copyright
```

The development version of `osmdata` can be installed with the `devtools` package using the following command:

```
devtools::install_github('osmdatar/osmdata')
```

Figure 18:

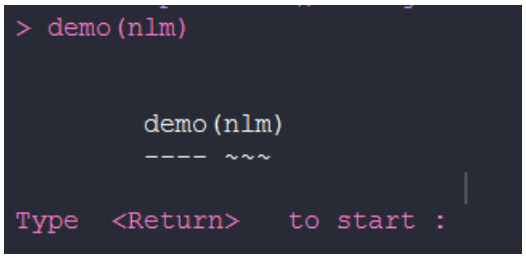


Figure 19:

```
##  [1] ".colMeans"       ".lm.fit"         "colMeans"
##  [4] "confint.lm"      "contr.helmert"   "dummy.coef.lm"
##  [7] "getAllMethods"   "glm"             "glm.control"
## [10] "glm.fit"         "KalmanForecast"  "KalmanLike"
## [13] "KalmanRun"       "KalmanSmooth"    "kappa.lm"
## [16] "lm"              "lm.fit"          "lm.influence"
## [19] "lm.wfit"         "model.matrix.lm" "nlm"
## [22] "nlminb"          "predict.glm"     "predict.lm"
## [25] "residuals.glm"   "residuals.lm"    "summary.glm"
## [28] "summary.lm"
```

- you can use that in combination with regular expressions

```
?"regular expression"
```

```
help.search("^glm")
```

- `??` is a synonym for `help.search`

Search engine for the R-Site

```
RSiteSearch("glm")
```

Usage of search engines

- I use **duckduckgo:**

```
R-project + "what I want to know"
```

- this works of course for all search engines!

Stackoverflow

- For questions about programming
- Is not focused on R - but **many discussions on R**
- Very detailed discussions

A cheatsheet for base R

https://www.rstudio.com/resources/cheatsheets/

More cheatsheets

Quick R

- Always a page with examples and help concerning a topic
- Example: **Quick R - Getting Help**

R Site Search

Query: glm Search! [How to search]

Display: 20 **Description:** normal **Sort:** by score

Target:

☑ Functions

☑ Task views

For problems WITH THIS PAGE (not with R) contact baron@upenn.edu.

Results:

References:

- **views**: [glm: 11]
- **vignettes**: [(can't open the index)]
- **functions**: [glm: 4391]

Total 4402 documents matching your query.

1. **R: Bias reduction in Binomial-response GLMs** (score: 299)
   Author: *unknown*
   Date: *Fri, 14 Jul 2017 10:27:38 -0500*
   Bias reduction in Binomial-response GLMs Description Usage Arguments Details Value Warnings brglm {brglm} R Documentation Fits bino
   http://finzi.psych.upenn.edu/R/library/brglm/html/brglm.html (21,462 bytes)

Figure 20:

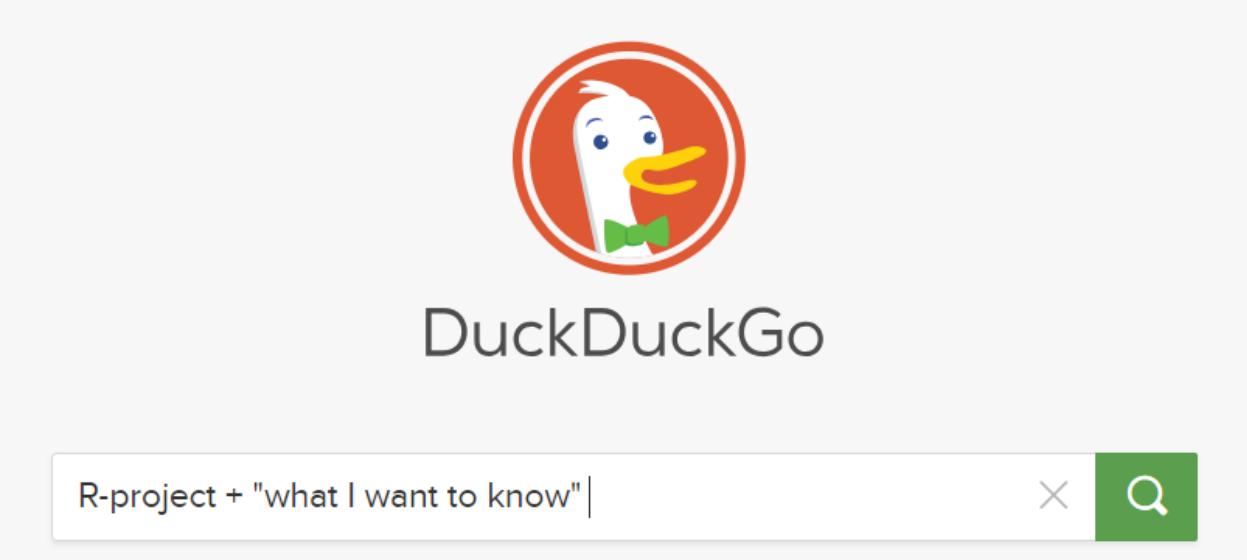


Figure 21:

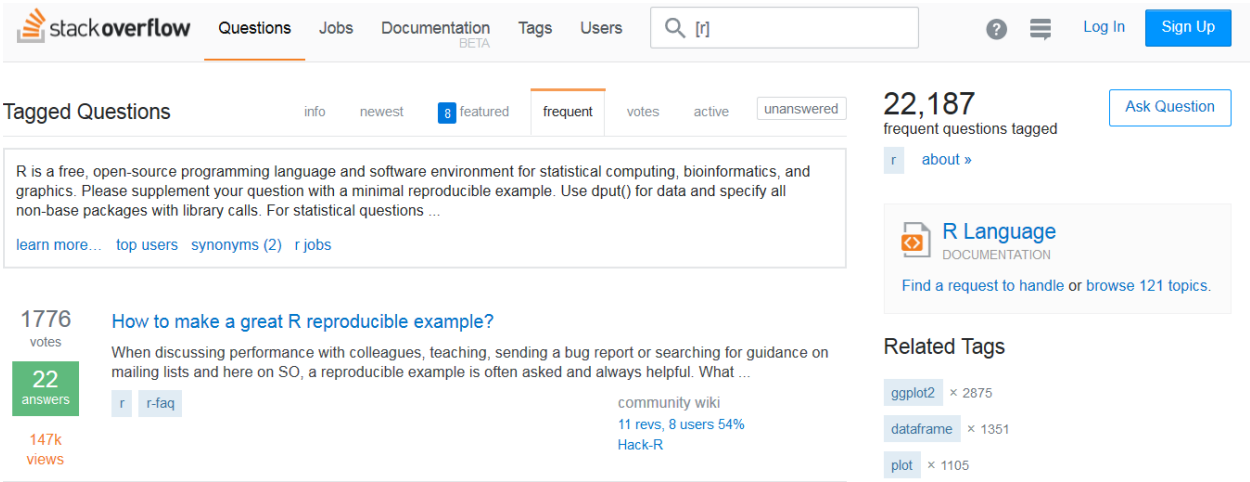


Figure 22: Stackoverflow Example

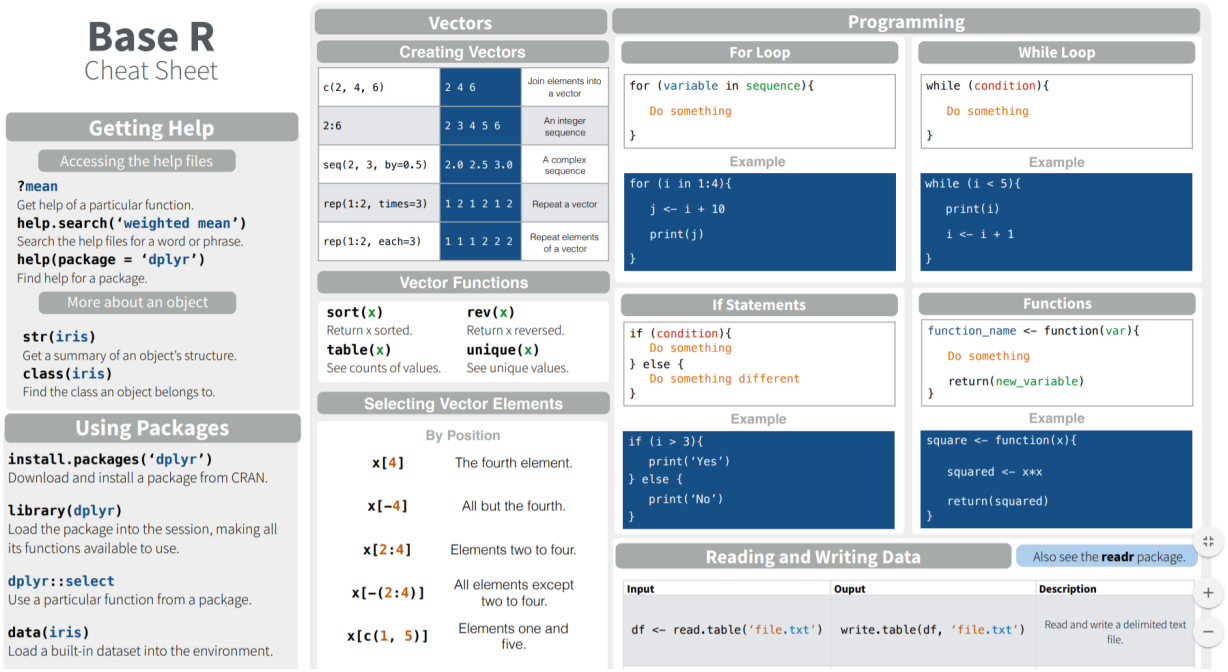


Figure 23: **Cheatsheet BaseR**

Further Links

- **Overview - how to get help in R**
- **A list with HowTo's**
- **A list with the most important R-commands**

Exercise A2A Getting help

- Try the command `?which.min` This opens a help page in the lower right window of RStudio. What does the function do?
- You must know the name of the function in order to open the help page as above. Sometimes (often, even) you do not know the name of the R functions; then a **search engine** can often help you. Try, for example, to search the text `R minimum vector`.

DataImport

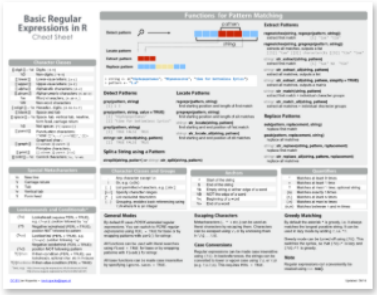
Data import

Import data with Rstudio

Rstudio functionality to import data

- Environment - Import Dataset - choose file type

Regular Expressions



Basics of regular expressions and pattern matching in R by Ian Kopacka. Updated 09/16.

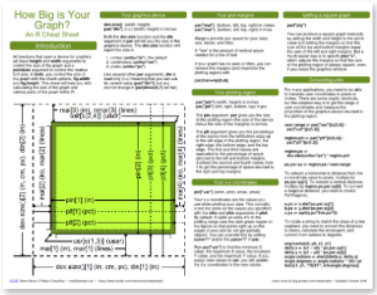
DOWNLOAD

The leaflet package

Interactive maps in R with leaflet, by Kejia Shi. Updated 05/17.

DOWNLOAD

How big is your graph?



Graph sizing with base R by by Stephen Simon. Updated 10/16.

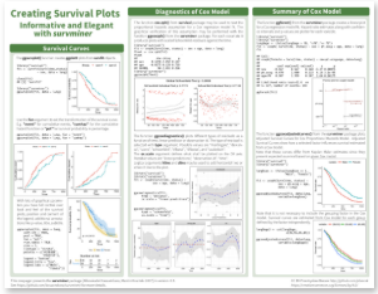
DOWNLOAD

The eurostat package

R tools to access the eurostat database, by rOpenGov. Updated 03/17.

DOWNLOAD

The survminer package



Elegant survival plots, by Przemyslaw Biecek. Updated 03/17.

DOWNLOAD

The sjmisc package

dplyr friendly Data and Variable Transformation, by Daniel Lüdecke. Updated 08/17.

DOWNLOAD

Figure 24:

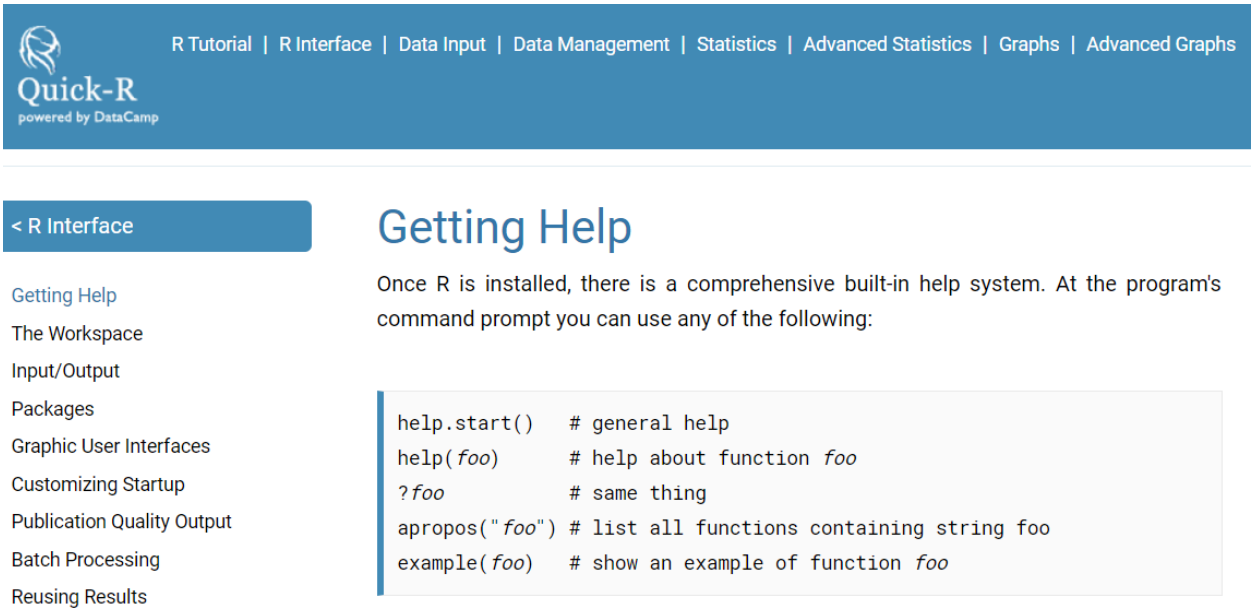


Figure 25:

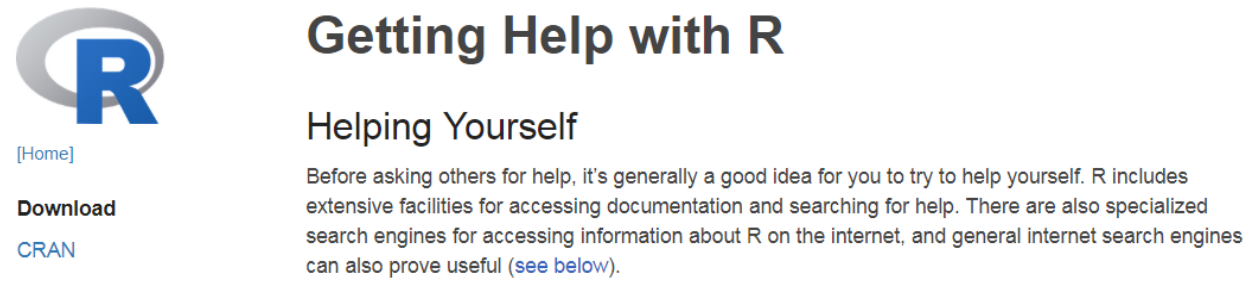


Figure 26:

Figure 27:

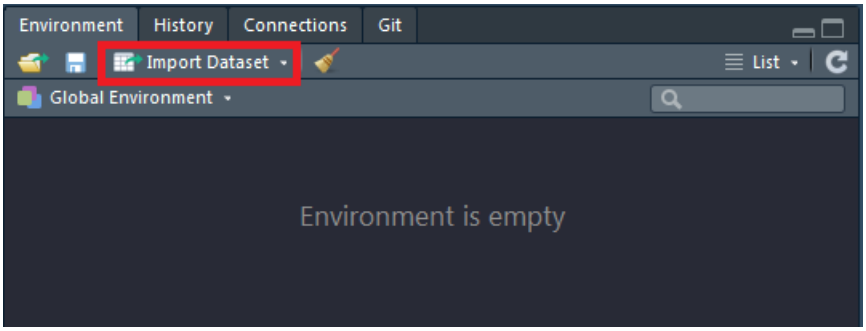


Figure 28:

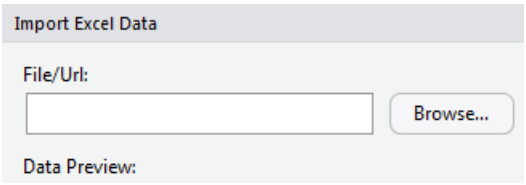


Figure 29:

Figure 30:

Where to find data

Browse Button in RStudio

Code preview in Rstudio

Import of csv data

- `read.csv` is a command available in base package
- Excel data can be saved as `.csv` in Excel
- Then `read.csv()` can be used to read in the data.
- For German data, you may need `read.csv2()` because of the comma separation.

```
dat <- read.csv("../data/ZA5666_v1-0-0.csv")
```

If it's German data:

```
datd <- read.csv2("../data/ZA5666_v1-0-0.csv")
```

The result - a `data.frame`

- the following `data.frame` is a small excerpt from the GESIS Panel data:

```
datd
```

```
##      z000001z z000002z          z000003z         z000005z
## 1   198431880    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 2   436122330    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 3   856844220    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 4   117346660    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 5   943433330    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 6   265582550    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 7   275587110    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 8   677771880    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 9   463671220    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 10 478839330    ZA5666 1-0-0 2017-06-20 10.4232/1.12749
```

Import Excel Dataset - with `xlsx`

Package `xlsx`

- Title: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files
- Authors: Adrian A. Dragulescu, Cole Arendt

```
install.packages("xlsx")
```

```
library("xlsx")
ab_xlsx <- read.xlsx("../data/ab.xlsx",1)
```

- The package `xlsx` needs Java - if that is not available use the command `read_excel` from package `readxl`

The package `readxl`

```
install.packages("readxl")
```

- **`readxl` has no external dependencies**
- `readxl` supports both the legacy `.xls` format and the modern xml-based `.xlsx` format.

```
library(readxl)
ab <- read_excel("../data/ab.xlsx")
head(ab)
```

```
## # A tibble: 4 x 3
##   X__1      a     b
##   <chr> <dbl> <dbl>
## 1 1         1     4
## 2 2         2     3
## 3 3         3     2
## 4 4         4     1
```

What is a `tibble`

- Tibbles are a modern take on data frames.
- **`tibble` is part of `tidyverse` (bundle of packages by Hadley Wickham)**

```
library(tibble)
as_tibble(ab_xlsx)
```

```
## # A tibble: 4 x 3
##   NA.       a     b
##   <fct> <dbl> <dbl>
## 1 1         1     4
## 2 2         2     3
## 3 3         3     2
## 4 4         4     1
```

Import SPSS files

- library `haven` - import and export 'SPSS', 'Stata' and 'SAS' files
- the result of this import command is a tibble

```
library(haven)
dataset <- read_sav("../data/ZA5666_v1-0-0.sav")
```

Import data from the web

Files can also be imported directly from the Internet:

```
library(foreign)
link <- "http://www.statistik.at/web_de/static/
mz_2013_sds_-_datensatz_080469.sav"

?read.spss
Dat <- read.spss(link,to.data.frame=T)
```

Import stata files

- With `read.dta13` stata files from version 13 can be imported

```
library(readstata13)
dat_stata <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
```

Import stata files - older versions

```
library(foreign)
dat_stata12 <- read.dta("../data/ZA5666_v1-0-0_Stata12.dta")
```

- Introduction to import with R (**is.R**)

The library readstata13

Import - GESIS Panel data

```
library(readstata13)
```

```
datf <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta",
                   convert.factors = F)
head(datf$bbzc007a)
```

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

For comparison - import without this argument

```
dat <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
head(dat$bbzc007a)
```

```
## [1] Nein         Nein         Nein         Nein         Ja, manchmal
## [6] Nein
## 10 Levels: Ambiguous answer Item nonresponse ... Ja, oft
```


readstata13 {readstata13} R Documentation

Import Stata Data Files

Description

Function to read the Stata file format into a data.frame.

Note

If you catch a bug, please do not sue us, we do not have any money.

Author(s)

Marvin Garbuszus jan.garbuszus@ruhr-uni-bochum.de

Sebastian Jeworutzki sebastian.jeworutzki@ruhr-uni-bochum.de

See Also

read.dta and memisc for dta files from Stata Versions < 13

Figure 31:

The argument `convert.factors = F`

```
?read.dta13
```

`convert.factors` - logical. If `TRUE`, factors from Stata value labels are created.

- `nonint.factors`

Get stata attributes

```
att_dat <- attributes(dat)
head(names(att_dat))
```

```
## [1] "row.names"  "names"      "datalabel"  "time.stamp" "formats"
## [6] "types"
```

Example: the variable names

```
head(att_dat$var.labels)
```

```
## [1] "Personen ID - Campus File"
## [2] "Studiennummer des Archivs"
## [3] "Versionskennung und -datum des Archivs"
## [4] "doi"
## [5] "Zufriedenheit Leben in Wohnort"
## [6] "Zufriedenheit Leben in Deutschland"
```

	z000001z Personen ID - Campus File	z000002z Studiennummer des Archivs	z000003z Versionskennung und -datum des Archivs	z000005z doi	a11c019a Zufriedenheit Leben in Wohnort	a11c020a Zufriedenheit
1	198431880	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
2	436122330	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
3	856844220	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	2	
4	117346660	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	
5	943433330	ZA5666	1-0-0 2017-06-20	10.4232/1.12749	1	

Figure 32: Rstudio Viewer

```
att_dat$names[att_dat$var.labels=="Zufriedenheit Leben in Wohnort"]
```

```
## [1] "a11c019a"
```

Get an initial overview of the data

```
View(datf)
```

- You can get the same in RStudio if you click on the dataset icon in the environment menue

The library `rio`

```
install.packages("rio")
```

```
library("rio")
x <- import("../data/ZA5666_v1-0-0.csv")
y <- import("../data/ZA5666_v1-0-0_Stata12.dta")
z <- import("../data/ZA5666_v1-0-0_Stata14.dta")
```

- **rio: A Swiss-Army Knife for Data I/O**

The package `Hmisc`

For SPSS and SAS I would recommend the Hmisc package for ease and functionality.

```
library(Hmisc)
mydata <- spss.get("c:/mydata.por", use.value.labels=TRUE)
# last option converts value labels to R factors
```

Import SAS data

```
mydata <- sasxport.get("c:/mydata.xpt")
# character variables are converted to R factors
```

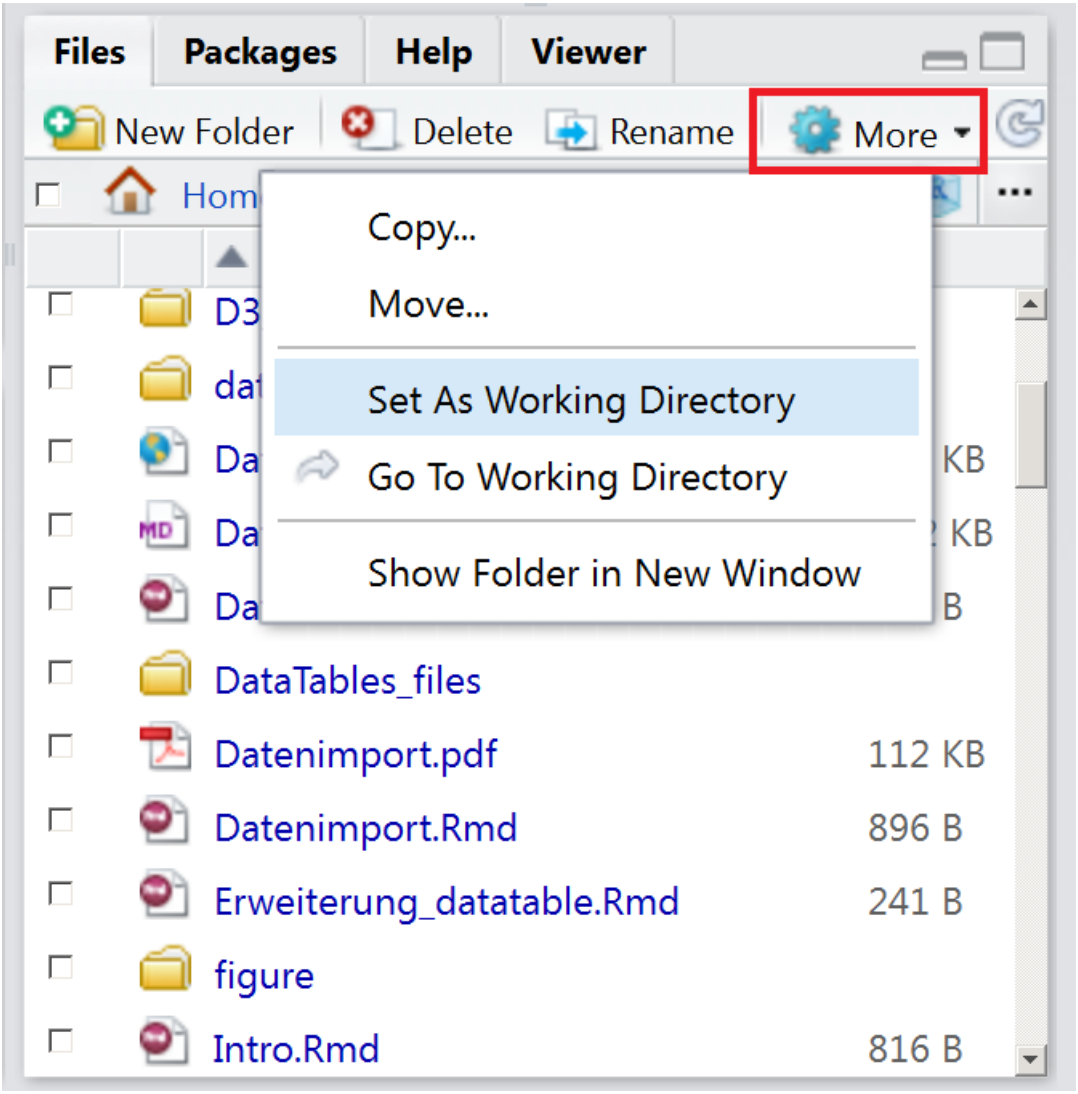

Files
Packages
Help
Viewer
New Folder
Delete
Rename
More
Hom
Copy...
Move...
Set As Working Directory
Go To Working Directory
Show Folder in New Window
DataTables_files
Datenimport.pdf
112 KB
Datenimport.Rmd
896 B
Erweiterung_datatable.Rmd
241 B
figure
Intro.Rmd
816 B

Figure 33:

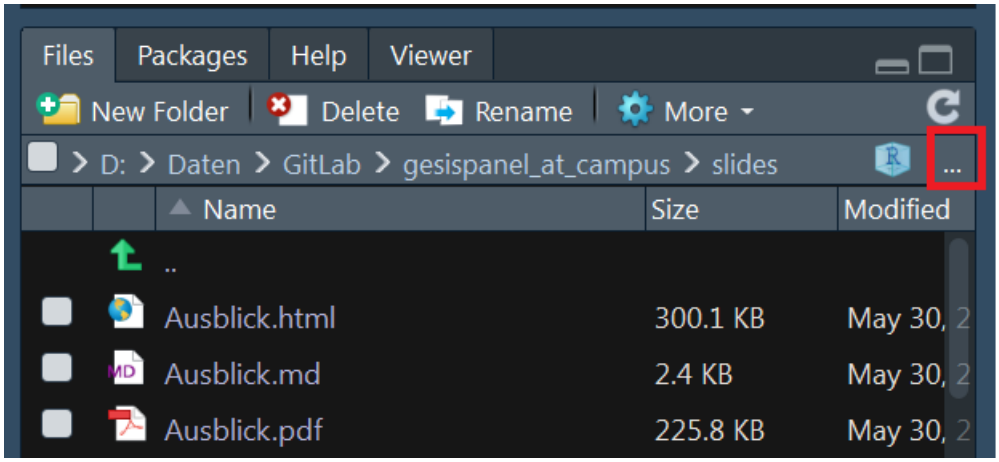


Figure 34:

The working directory

...

- If the data is on a different drive in Windows

The working directory II

This way you can find out which directory you are currently in

```
getwd()
```

So you can change the working directory:

You create an object in which you save the path:

```
main.path <- "C:/" # Example for Windows
main.path <- "/users/Name/" # Example for Mac
main.path <- "/home/user/" # Example for Linux
```

And then change the path with `setwd()`

```
setwd(main.path)
```

On Windows it is important to use slashs instead of backslashes.

Change working directory

- You can also use the tab key to get the autocompletion.

```
getwd()
```

```
## [1] "D:/gitlab/IntroDataAnalysis/slides"
```

```
setwd("..")
getwd()
```

```
## [1] "D:/gitlab/IntroDataAnalysis"
```

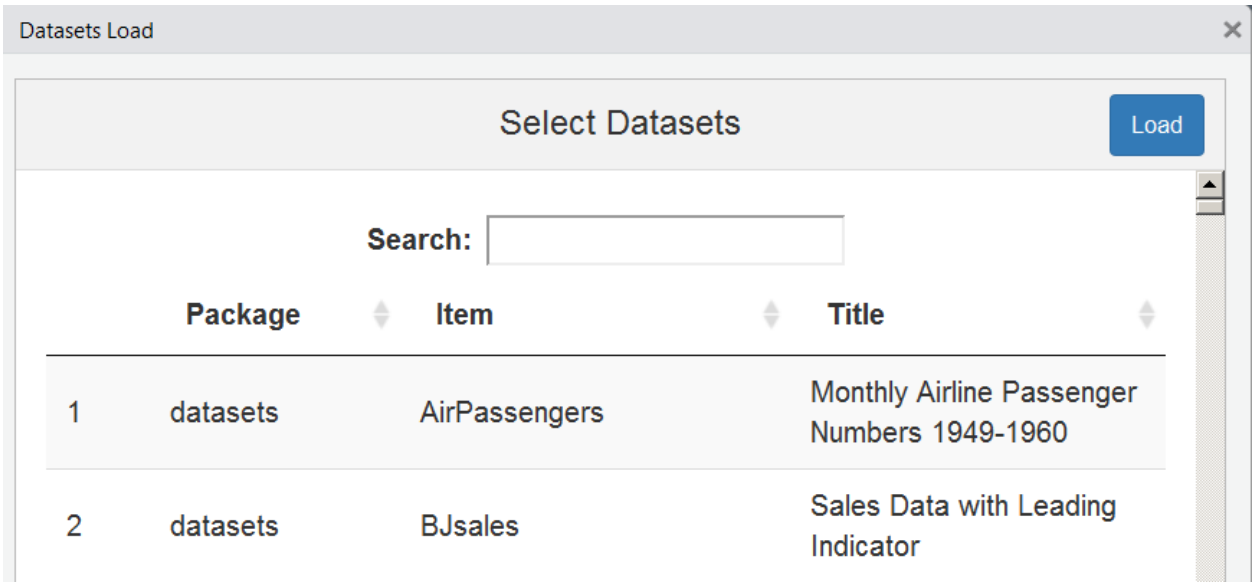


Figure 35:

Built-In datasets

- Often an example dataset is provided to show the functionality of a package
- These datasets can be loaded with the command `data`

```
data(iris)
```

- There is also an **RStudio add-in** that helps to find a dataset

```
install.packages("datasets.load")
```

Inserting data

- **RStudio addin for inserting data**

```
devtools::install_github("lbusett/insert_table")
```

A4A Exercise - Import the GESIS Panel data

- Please import the GESIS panel data with a suitable command

Data Export

Creating an example data record

```
A <- c(1,2,3,4)
B <- c("A","B","C","D")

mydata <- data.frame(A,B)
```

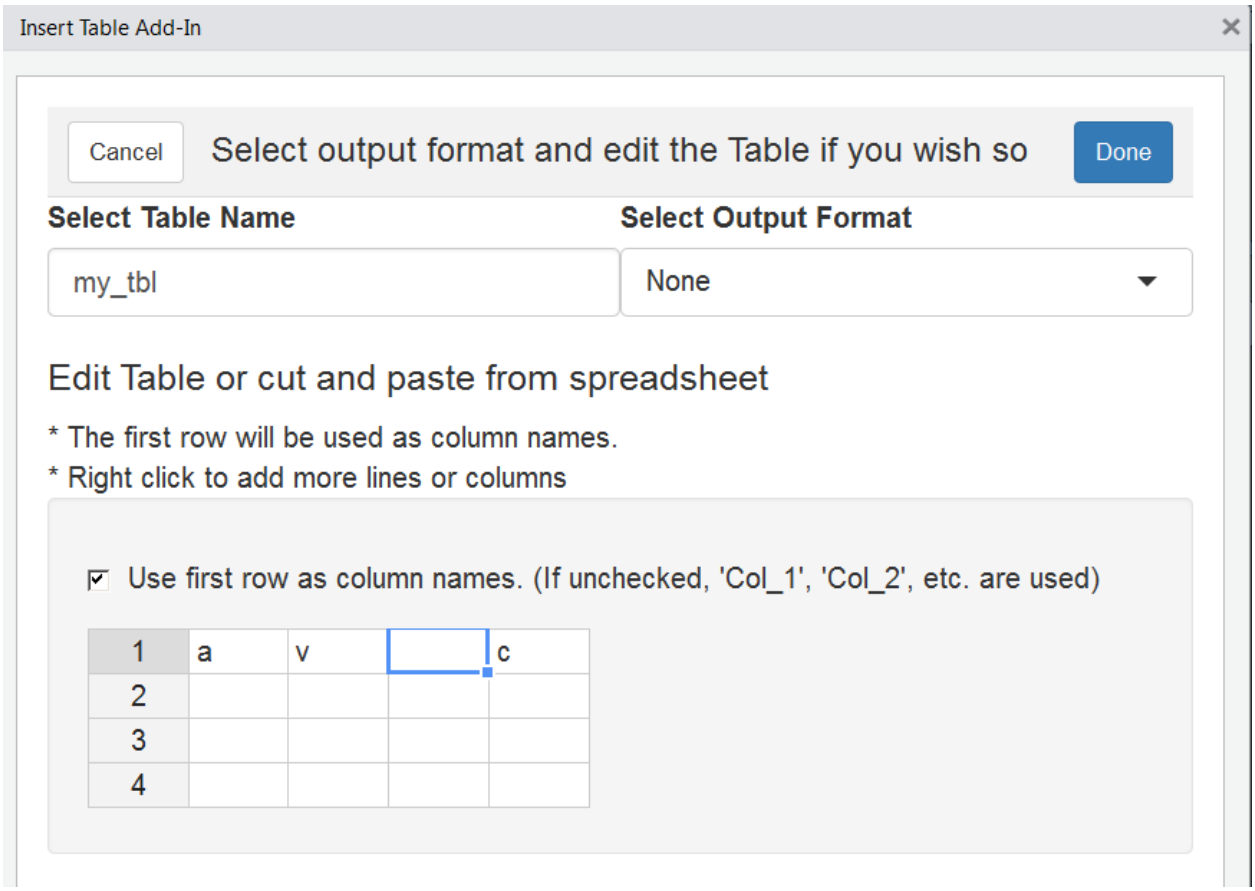

Insert Table Add-In
Cancel
Select output format and edit the Table if you wish so
Done
Select Table Name
Select Output Format
my_tbl
None
Edit Table or cut and paste from spreadsheet
* The first row will be used as column names.
* Right click to add more lines or columns
Use first row as column names. (If unchecked, 'Col_1', 'Col_2', etc. are used)
1 a v c
2
3
4

Figure 36:

```
mydata
```

A	B
1	A
2	B
3	C
4	D

Overview data import/export

- if you continue working with R, `.RData` or **`rds`** format is the best choice:

```
save(mydata, file="mydata.RData")
saveRDS(mydata, "mydata.rds")
```

- The data set can be imported with `load`.

```
load("mydata.RData")
mydata <- readRDS("mydata.rds")
```

- `saveRDS()` doesn't save the both the object and its name it just saves a representation of the object

Export as Excel

- Create a example tibble:

```
library(tibble)
ab <- tibble(a=1:4,b=4:1)
```

```
library(xlsx)
setwd("D:/Daten/GitLab/IntroDataAnalysis/data")
write.xlsx(ab,file="ab.xlsx")
```

Addin to open dataset in Excel

```
devtools::install_github("dreamRs/viewxl")
```

- select a `data.frame` in script -> it is opened in Excel.

Save data in `.csv` format

```
write.csv(mydata,file="mydata.csv")
```

- If you want to continue working with German Excel, it is better to use `write.csv2`

```
write.csv2(mydata,file="mydata.csv")
```

- Otherwise, the result looks like this:

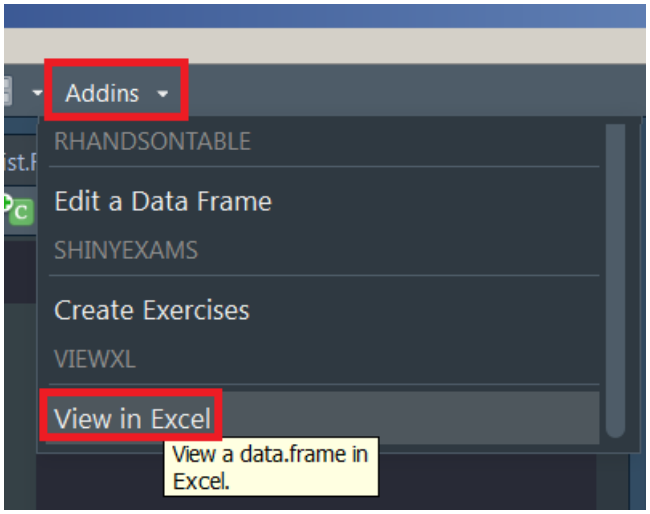


Figure 37:

	A
1	,"A","B"
2	1,1,"A"
3	2,2,"B"
4	3,3,"C"
5	4,4,"D"
6	

Figure 38:

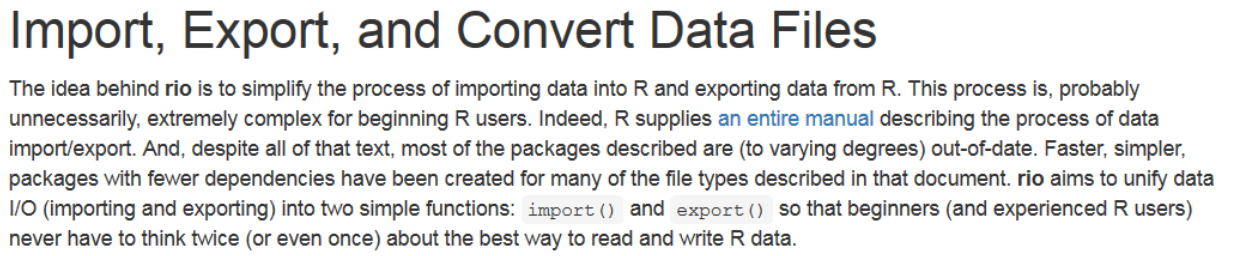


Figure 39:

Argument `row.names`

- ** Prevent row names to be written to file when using write.csv**

```
write.csv(mydata,file="mydata.csv", row.names=FALSE)
```

The package `rio`

```
install.packages("rio")
```

Save data as `.sav` (SPSS)

```
library("rio")
# create file to convert

export(mtcars, "data/mtcars.sav")
```

Convert file formats

```
export(mtcars, "data/mtcars.dta")

# convert Stata to SPSS
convert("data/mtcars.dta", "data/mtcars.sav")
```

Links Export

- **Quick R** for the export of data
- Help for exporting on the **CRAN Server**
- **Export data from R**

A5A Exercise - Export dataset

- Please load the `iris` example dataset
- Export the `iris` dataset to Excel

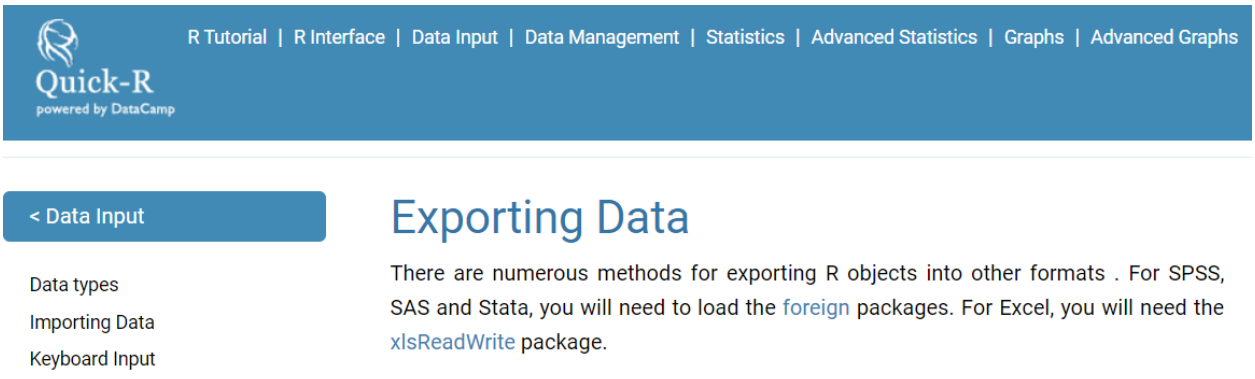


Figure 40:

Data Processing

Data Frames

- Import example data:

```
library("readstata13")
dat <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta")
```

```
typeof(dat)
```

```
## [1] "list"
```

```
head(names(dat))
```

```
## [1] "z000001z" "z000002z" "z000003z" "z000005z" "a11c019a" "a11c020a"
```

Transfer to dataframe

- Combine these two vectors to a `data.frame`:

```
gpdat <- data.frame(dat)
```

- Find out the number of rows/columns

```
nrow(gpdat) # rows
```

```
## [1] 100
```

```
ncol(gpdat) # columns
```

```
## [1] 1192
```

View the data

- See the some lines:

```
head(gpdat) # first lines
tail(gpdat) # last lines
```

- Overview with Rstudio:

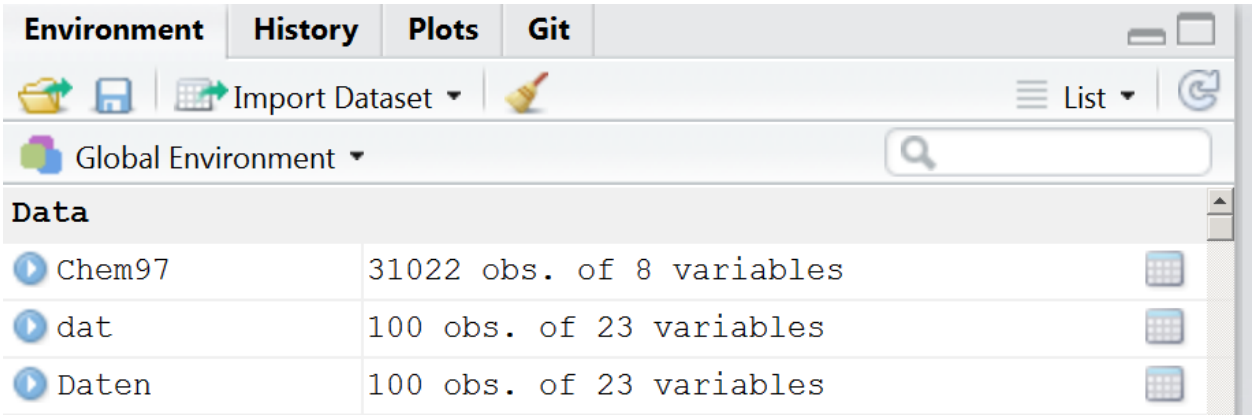


Figure 41:

Indexing a `data.frame`

- Example dataset:

```
load("../data/mydata.RData")
```

```
mydata[1,1] # getting the element at the top left
```

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

```
mydata[2,] # see only the second row
```

```
##   A B
## 2 2 B
```

```
mydata[,1] # see only the first column
```

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

Indexing a `data.frame` II

```
mydata[1:2,] # getting the first two rows
```

```
##   A B
## 1 1 A
## 2 2 B
```

Indexing

- The dollar sign can also be used to address individual columns

```
head(datf$a11c019a)
```

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

```
datf$a11c019a[1:10]
```

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

Accessing Columns

- As already described, you can also use numbers to access the columns

```
head(datf[,1])
head(datf[,"a11c019a"]) # the same result
```

GESIS Panel Variable - Estimated duration (bazq020a)

How long did it take you to fill in the questionnaire?

```
duration <- as.numeric(datf$bazq020a)
```

```
summary(duration)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -99.00   10.00   15.00   25.57   20.00 1440.00       2
```

Recode missing values

```
gpdat$bazq020a[gpdat$bazq020a==-99] <- NA
```

- **Quick-R about missing values**
- **Recode missing values**

Writing a simple function

```
tail(duration,n=10)
```

```
##  [1]  30  25   6  15  12  15  10  10 -33  15
```

```
transform_miss <- function(x){
  x[x%in%c(-11,-22,-33,-44,-55,-66,-77,-88,-99,-111)] <- NA
  return(x)
}
```

```
duration <- transform_miss(duration)
tail(duration,n=10)
```

```
##  [1] 30 25  6 15 12 15 10 10 NA 15
```

Missing values

- Missing values are defined as `NA` in R
- Math functions usually have a way to exclude missing values in their calculations.
- `mean()`, `median()`, `colSums()`, `var()`, `sd()`, `min()` und ‘max() all take the na.rm argument.

```
mean(duration)
```

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

```r
mean(duration,na.rm=T)
```

```
## [1] 34.24138
```

The command `complete.cases()`

- The function `complete.cases()` returns a logical vector indicating which cases are complete.

```r
# list rows of data that have missing values
mydata[!complete.cases(mydata),]
```

Specify different types of missing values (NAs)

- With the package `memisc` you can specify different types of missing values
- Use the function `include.missings()` to do so

```r
library(memisc)
?include.missings
```

- It is also possible to create codebook entries with that package.

```r
codebook(dat$a11c019a)
```

Restrict with the `tidyverse` package

```r
head(datf[duration>20,1:4])
```

```
##     z000001z z000002z         z000003z        z000005z
## 11 854241220   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 15 369554550   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## NA        NA     <NA>             <NA>            <NA>
## 21 567537770   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 25 496531440   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
## 27 783874440   ZA5666 1-0-0 2017-06-20 10.4232/1.12749
```

- the same with a command from `tidyverse` package

```r
library(tidyverse)
filter(datf, duration>20)
```

Subsetting dataset

```r
SEX <- gpdat$a11d054a
table(SEX)
```

```
## SEX
## Männlich Weiblich
##       44       56
```

```r
gpdat[SEX=="Männlich",]
# same result:
gpdat[SEX!="Weiblich",]
```

Other important options

```
# Save result to an object
subDat <- gpdat[duration>20,]
```

```
# multiple conditions can be
# linked with &
gpdat[duration>18 & SEX=="Männlich",]
```

The use of sequences in indexing

```
library("readstata13")
datf <- read.dta13("../data/ZA5666_v1-0-0_Stata14.dta",
                   convert.factors = F)
```

```
datf[15:23,10:14]
```

```
##    a11c024a a11c025a a11c026a a11c027a a11c028a
## 15        1        2        5        5        1
## 16        3        2        4        1        1
## 17        1        1        5        2        4
## 18        2        2        3        3        1
## 19        2        1        4        5        1
## 20        1        2        3        2        1
## 21        2        2        5        5        1
## 22        1        3        4        3        2
## 23        2        2        2        3        1
```

Variable labels

```
library(foreign)
dat <- read.dta("../data/ZA5666_v1-0-0_Stata12.dta")
```

```
attributes(dat)
```

```
var.labels <- attr(dat,"var.labels")
```

- The same applies to the `haven` package

```
library(haven)
dat2 <- read_dta("../data/ZA5666_v1-0-0_Stata14.dta")
var.labels2 <- attr(dat,"var.labels")
```

Rename the column names

- With the command `colnames` you get the column names

```
colnames(dat)
```

- We can rename the column names:

```
colnames(dat) <- var.labels
```

- The same applies to the row names

```r
rownames(dat)
```

Excursus - how to use labels

Tools for Working with Categorical Variables (Factors)

```r
library("forcats")
```

- `fct_collapse` - to summarize factor levels
- `fct_count` - to count the entries in a factor
- `fct_drop` - Take out unused levels

The command `fct_count`

```r
fct_count(f = dat$a11c026a)
```

```
## # A tibble: 8 x 2
##   f                              n
##   <fct>                      <int>
## 1 Item nonresponse               0
## 2 Täglich                       23
## 3 Mehrmals die Woche            19
## 4 Mehrmals im Monat             12
## 5 Mindestens einmal im Monat     9
## 6 Seltener                      26
## 7 Nie                           11
## 8 Weiß nicht                     0
```

The command `fct_collapse`

```r
a11c026a <- fct_collapse(.f = dat$a11c026a,
    Mehrmals=c("Mehrmals die Woche","Mehrmals im Monat"))
```

```r
fct_count(a11c026a)
```

```
## # A tibble: 7 x 2
##   f                              n
##   <fct>                      <int>
## 1 Item nonresponse               0
## 2 Täglich                       23
## 3 Mehrmals                      31
## 4 Mindestens einmal im Monat     9
## 5 Seltener                      26
## 6 Nie                           11
## 7 Weiß nicht                     0
```

`recode` command in package `car`

```r
library(car)
```

```r
head(dat$a11c020a)
```

```
## [1] Sehr zufrieden Sehr zufrieden Eher zufrieden Sehr zufrieden
## [5] Sehr zufrieden Sehr zufrieden
## 7 Levels: Item nonresponse Sehr zufrieden ... Weiß nicht
```

```r
head(recode(dat$a11c020a,"'Eher unzufrieden'='A';else='B'"))
```

```
## [1] B B B B B B
## Levels: A B
```

Loops in R

- the command `for` indicates the start of a loop
- in brackets, we have a index and the number of runs (in this case the loop runs from 1 until the number of columns in dat)
- in the curly breakets it is speciefied what happens for one iteration

```r
for (i in 1:ncol(dat)){
  dat[,i] <- as.character(dat[,i])
}
```

Loops - keeping results

- we can save the results in a container
- that can be a vector or a list

```r
erg1 <- vector()
erg2 <- list()

for (i in 1:ncol(dat)){
  tab <- table(dat[,i])
  erg[i] <- length(tab)
  erg[[i]] <- tab
  cat(i, "\n")
}
```

The apply family

Example data set

```r
ApplyDat <- cbind(1:4,runif(4),rnorm(4))
ApplyDat
```

```
##      [,1]       [,2]       [,3]
## [1,]    1 0.61255118 -0.7628428
## [2,]    2 0.90861015  0.5102232
## [3,]    3 0.47701986 -0.7135488
## [4,]    4 0.02752626  0.7439935
```

```r
apply(ApplyDat,1,mean)
```

```
## [1] 0.2832361 1.1396111 0.9211570 1.5905066
```

```r
apply(ApplyDat,2,mean)
```

```
## [1]  2.50000000  0.50642686 -0.05554375
```

The command apply()

```r
apply(ApplyDat,1,var)
```

```
## [1] 0.8582400 0.5948798 3.5955545 4.4825752
```

```r
apply(ApplyDat,1,sd)
```

```
## [1] 0.9264124 0.7712845 1.8961948 2.1172093
```

```r
apply(ApplyDat,1,range)
```

```
##            [,1]      [,2]       [,3]       [,4]
## [1,] -0.7628428 0.5102232 -0.7135488 0.02752626
## [2,]  1.0000000 2.0000000  3.0000000 4.00000000
```

The arguments of the command apply()

- If `margin=1` the function `mean` is applied for rows,
- If `margin=2` the function `mean` is applied for columns,
- Instead of `mean` you could also use `var`, `sd` or `length`.

The command tapply()

```r
ApplyDat <- data.frame(Income=rnorm(5,1400,200),
                       Sex=sample(c(1,2),5,replace=T))
```

Example command tapply()

```r
tapply(ApplyDat$Income,
       ApplyDat$Sex,function(x)x)
```

```
## $`1`
## [1] 1492.946 1649.956
##
## $`2`
## [1] 1342.477 1309.182 1013.895
```

- Other functions can also be used.....
- Self-programmed functions

B1A Exercise - using the tapply() command

- Find out which variable contains information about age
- Calculate the average duration (variable `bfzq020a`) by age group

```
##      Ambiguous answer      Item nonresponse           Not reached
##                    NA              21.45455                   NaN
##      Unit nonresponse          Not in panel 18 bis unter 20 Jahre
##                   NaN                   NaN              19.07692
## 20 bis unter 25 Jahre 25 bis unter 30 Jahre 30 bis unter 35 Jahre
##              15.03125              16.55224              20.54412
## 35 bis unter 40 Jahre 40 bis unter 45 Jahre 45 bis unter 50 Jahre
##              19.21818              22.40789              20.61475
## 50 bis unter 55 Jahre 55 bis unter 60 Jahre 60 bis unter 63 Jahre
##              20.86364              20.35238              25.88889
## 63 bis unter 65 Jahre 65 bis unter 70 Jahre    70 Jahre und Älter
##              23.87500              23.27907              24.81081
```

The reshape package

The function melt

```
# sample dataset
mydata <- data.frame(id=rep(1:2,each=2),
                     time=rep(c(1,2),2),
                     x1 = c(5,3,6,2),
                     x2 = c(6,5,1,4))
mydata
```

```
##   id time x1 x2
## 1  1    1  5  6
## 2  1    2  3  5
## 3  2    1  6  1
## 4  2    2  2  4
```

```
# example of melt function
library(reshape)
melt(mydata, id=c("id","time"))
```

```
##   id time variable value
## 1  1    1       x1     5
## 2  1    2       x1     3
## 3  2    1       x1     6
## 4  2    2       x1     2
## 5  1    1       x2     6
## 6  1    2       x2     5
## 7  2    1       x2     1
## 8  2    2       x2     4
```

The package tibble

```
install.packages("tibble")
```

```
library(tibble)
gpanel1 <- as_tibble(dat)
gpanel1
```

```
## # A tibble: 1,222 x 1,193
```

```
##    z000001z z000002z z000003z z000005z a11c019a a11c020a a11c021a a11c022a
##  *    <int> <chr>    <chr>    <chr>    <fct>    <fct>    <fct>    <fct>
##  1   1.98e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Sehr zu~ Stimme ~
##  2   4.36e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Sehr zu~ Stimme ~
##  3   8.57e8 ZA5666   1-0-0 2~ 10.4232~ Eher zu~ Eher zu~ Eher zu~ Stimme ~
##  4   1.17e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Sehr zu~ Stimme ~
##  5   9.43e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Eher un~ Stimme ~
##  6   2.66e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Sehr zu~ Stimme ~
##  7   2.76e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Weder z~ Stimme ~
##  8   6.78e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Eher zu~ Sehr zu~ Stimme ~
##  9   4.64e8 ZA5666   1-0-0 2~ 10.4232~ Eher zu~ Eher zu~ Sehr zu~ Stimme ~
## 10   4.79e8 ZA5666   1-0-0 2~ 10.4232~ Sehr zu~ Sehr zu~ Eher zu~ Stimme ~
## # ... with 1,212 more rows, and 1,185 more variables: a11c023a <fct>,
## #   a11c024a <fct>, a11c025a <fct>, a11c026a <fct>, a11c027a <fct>,
## #   a11c028a <fct>, a11c029a <fct>, a11c030a <fct>, a11c031a <fct>,
## #   a11c032a <fct>, a11c033a <fct>, a11c034a <fct>, a11c035a <fct>,
## #   a11c036a <fct>, a11c037a <fct>, a11c038a <fct>, a11c039a <fct>,
## #   a11c040a <fct>, a11c041a <fct>, a11c042a <fct>, a11c043a <fct>,
## #   a11c044a <fct>, a11c045a <fct>, a11c046a <fct>, a11c047a <fct>,
## #   a11c048a <fct>, a11c049a <fct>, a11c050a <fct>, a11c051a <fct>,
## #   a11c052a <fct>, a11c053a <fct>, a11d054a <fct>, a11d056z <fct>,
## #   a11d057d <fct>, a11d072d <fct>, a11d074b <fct>, a11d075d <fct>,
## #   a11d077d <fct>, a11d079b <fct>, a11d080a <fct>, a11d081a <fct>,
## #   a11d082b <fct>, a11d086b <fct>, a11d089c <fct>, a11d090b <fct>,
## #   a11d092a <fct>, a11d093b <fct>, a11d094a <fct>, a11d095b <fct>,
## #   a11d096b <fct>, a11d097c <fct>, a11c098a <fct>, a11c099a <fct>,
## #   a11c100a <fct>, a11c101a <fct>, a11c102a <fct>, a11c103a <fct>,
## #   a11c104a <fct>, a11c105a <fct>, a11c106a <fct>, a11c107a <fct>,
## #   a11c108a <fct>, a11c109a <fct>, a11c110a <fct>, a11c111a <fct>,
## #   a11c112a <fct>, a11c113a <fct>, a11c114a <fct>, a11c115a <fct>,
## #   a11c116a <fct>, a11c117a <fct>, a11c118a <fct>, a11c119a <fct>,
## #   a11c132a <fct>, a11a167a <fct>, a11a169b <fct>, a12c001a <fct>,
## #   a12c002a <fct>, a12c003a <fct>, a12c004a <fct>, a12c005a <fct>,
## #   a12c006a <fct>, a12c007a <fct>, a12c008a <fct>, a12c009a <fct>,
## #   a12c010a <fct>, a12c011a <fct>, a12c012a <fct>, a12c013a <fct>,
## #   a12c014a <fct>, a12c015a <fct>, a12c016a <fct>, a12c017a <fct>,
## #   a12c018a <fct>, a12c019a <fct>, a12c020a <fct>, a12d021b <fct>,
## #   a12c022b <fct>, a12c023a <fct>, a12c025a <fct>, ...
```

Difference between tibble and data.frame

- There are three key differences between tibbles and data frames: printing, subsetting, and recycling rules.

Further Links

- Tidy data - the package `tidyr`
- The `tidyverse` collection
- Data wrangling with R and RStudio
- Hadley Wickham - Tidy Data

- Hadley Wickham - Advanced R
- Colin Gillespie and Robin LovelaceEfficient R programming

Lattice

A plot says more than 1000 words

- Graphical data analysis is great
- Good plots can contribute to a better understanding
- Generating a plot is easy
- Making a good plot can take a very long
- Generating plots with R is fun
- Plots created with R have high quality
- Almost every plot type is supported by R
- A large number of export formats are available in R

Not all plots are the same

- The base package already includes a large number of plot functions
- Other packages like `lattice`, `ggplot2`, etc extend this functionality

Manuals that go far beyond this introduction:

- Murrell, P (2006): R Graphics.
- R Development Core Group **Graphics with R**
- Wiki on **R Programming/Graphics**
- Martin Meermeyer **Creating Reproducible Publication Quality Graphics with R: A Tutorial**
- Institute For Quantitative Social Science at Harvard - **R graphics tutorial**

Task View for graphics

CRAN Task View: Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization

Maintainer: Nicholas Lewin-Koh
Contact: nikko at hailmail.net
Version: 2015-01-07
URL: https://CRAN.R-project.org/view=Graphics

R is rich with facilities for creating and developing interesting graphics. Base R contains functionality for many plot types including coplots, mosaic plots, biplots, and the list goes on. There are devices such as postscript, png, jpeg and pdf for outputting graphics as well as device drivers for all platforms running R. lattice and grid are supplied with R's recommended packages and are included in every binary distribution. lattice is an R implementation of William Cleveland's trellis graphics, while grid defines a much more flexible graphics environment than the base R graphics.

R's base graphics are implemented in the same way as in the S3 system developed by Becker, Chambers, and Wilks. There is a static device, which is treated as a static canvas and objects are drawn on the device through R plotting commands. The device has a set of global parameters such as margins and layouts which can be manipulated by the user using `par()` commands. The R graphics engine does not maintain a user visible graphics list, and there is no system of double buffering, so objects cannot be easily edited without redrawing a whole plot. This situation may change in R 2.7.x, where developers are working on double buffering for R devices. Even so, the base R graphics can produce many plots with extremely fine graphics in many specialized instances.

One can quickly run into trouble with R's base graphic system if one wants to design complex layouts where scaling is maintained properly on resizing, nested graphs are desired or more interactivity is needed. grid was designed by Paul Murrell to overcome some of these limitations and as a result packages like lattice, ggplot2, vcd or hexbin use grid for the underlying primitives. When using plots designed with grid one needs to keep in mind that grid is based on a system of viewports and graphic objects. To add objects one needs to use grid commands, e.g., `grid.polygon()` rather than `polygon()`. Also grid maintains a stack of viewports from the device and one needs to make sure the desired viewport is at the top of the stack. There is a great deal of explanatory documentation included with grid as vignettes.

The graphics packages in R can be organized roughly into the following topics, which range from the more user oriented at the top to the more developer oriented at the bottom. The categories are not mutually exclusive but are for the convenience of presentation:

https://cran.r-project.org/web/views/Graphics.html

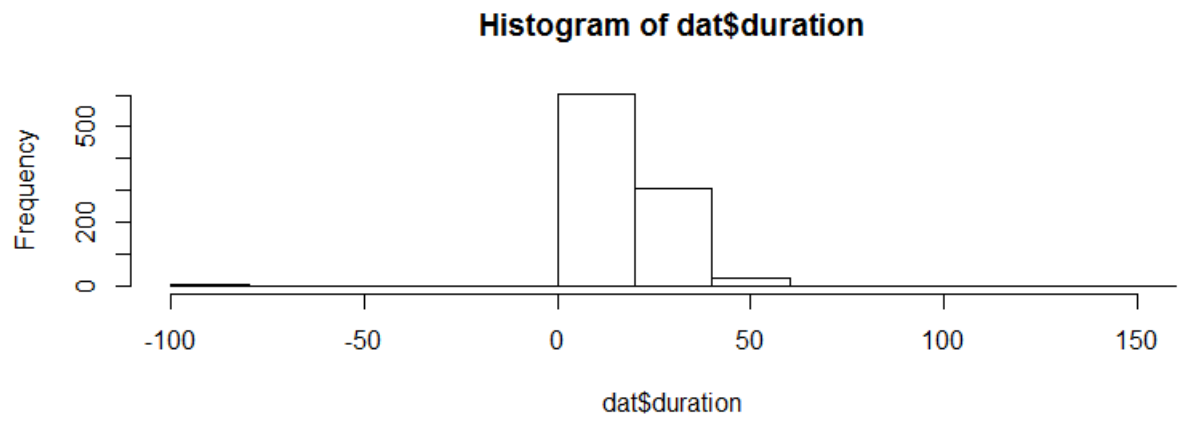


Figure 42:

Histogram - The `hist()` function

We create a histogram of the variable `duration`:

```
?hist
```

```
hist(dat$duration)
```

Export with Rstudio

Command to save graphic

- Alternatively also with the commands `png`, `pdf` or `jpeg` for example

```
png("Histogramm.png")
  hist(dat$duration)
dev.off()
```

```
pdf("Histogramm.pdf")
  hist(dat$duration)
dev.off()
```

```
jpeg("Histogramm.jpeg")
  hist(dat$duration)
dev.off()
```

Histogram

- Command `hist()` plots a histogram
- At least one observation vector must be passed to the function
- `hist()` has many more arguments, which all have (meaningful) default values

```
hist(dat$duration,col="blue",
     main="Duration of interview",ylab="Frequency",
     xlab="Duration")
```

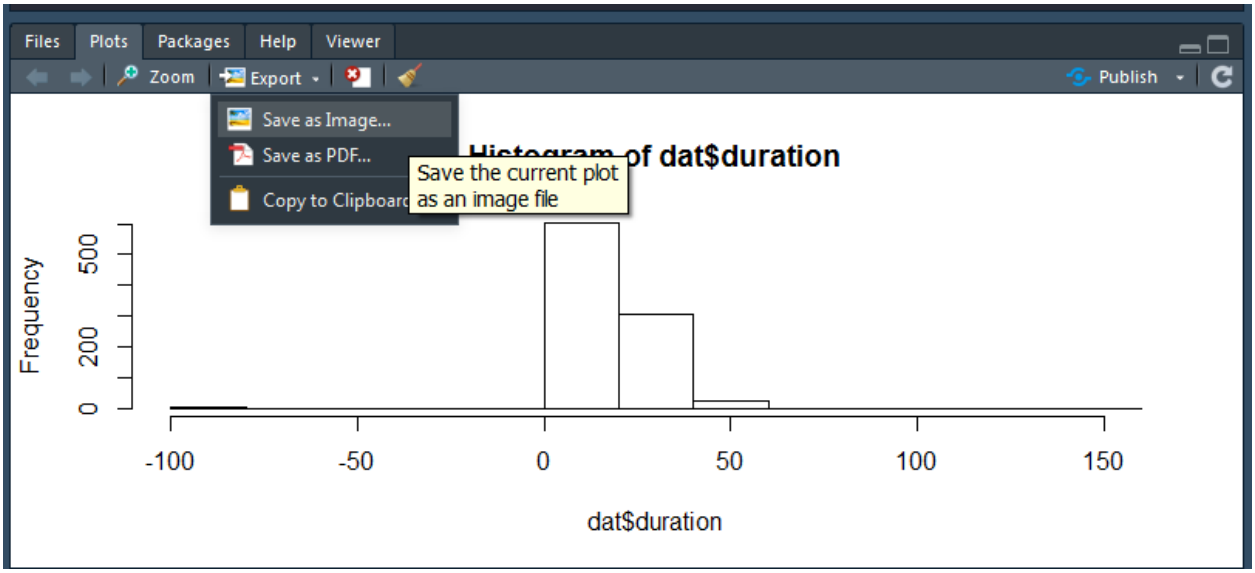
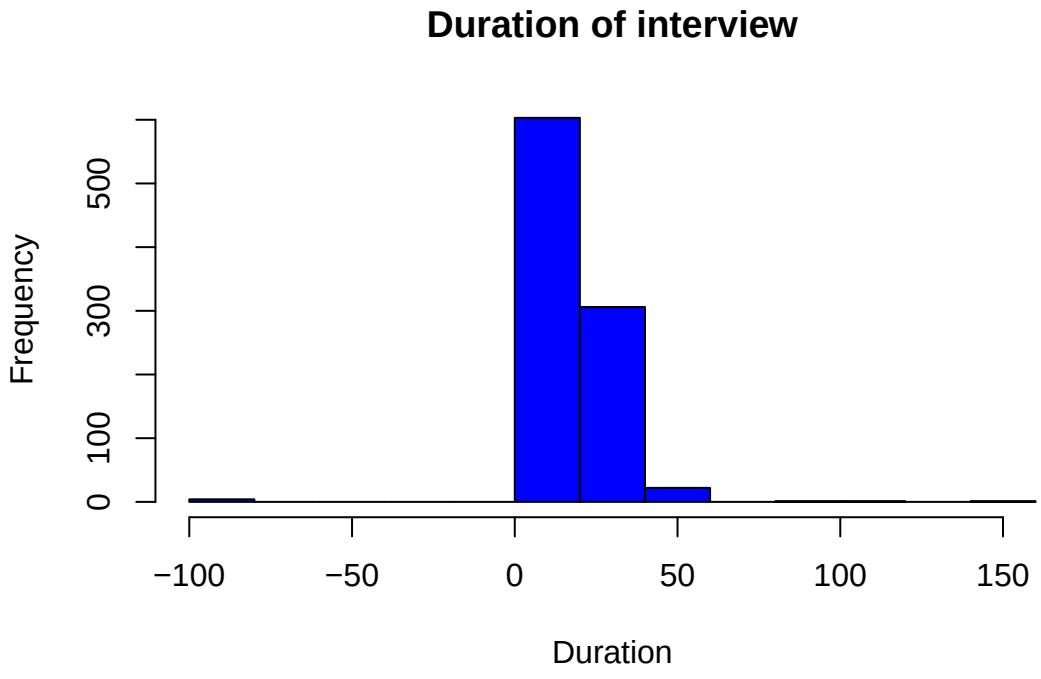


Figure 43:

Histogram



Further arguments:

```
?plot
# or
```

Graphical Parameters

adj

The value of adj determines the way in which text strings are justified in text, mtext and title. A value of 0 produces left-justified text, 0.5 (the default) centered text and 1 right-justified text. (Any value in *[0, 1]* is allowed, and on most devices values outside that interval will also work.)

Note that the adj *argument* of text also allows adj = c(x, y) for different adjustment in x- and y- directions. Note that whereas for text it refers to positioning of text about a point, for mtext and title it controls placement within the plot or device region.

ann

If set to FALSE, high-level plotting functions calling plot.default do not annotate the plots they produce with axis titles and overall titles. The default is to do annotation.

ask

logical. If TRUE (and the R session is interactive) the user is asked for input, before a new figure is drawn. As this applies to the device, it also affects output by packages **grid** and **lattice**. It can be set even on non-screen devices but may have no effect there.

This not really a graphics parameter, and its use is deprecated in favour of devAskNewPage.

Figure 44:

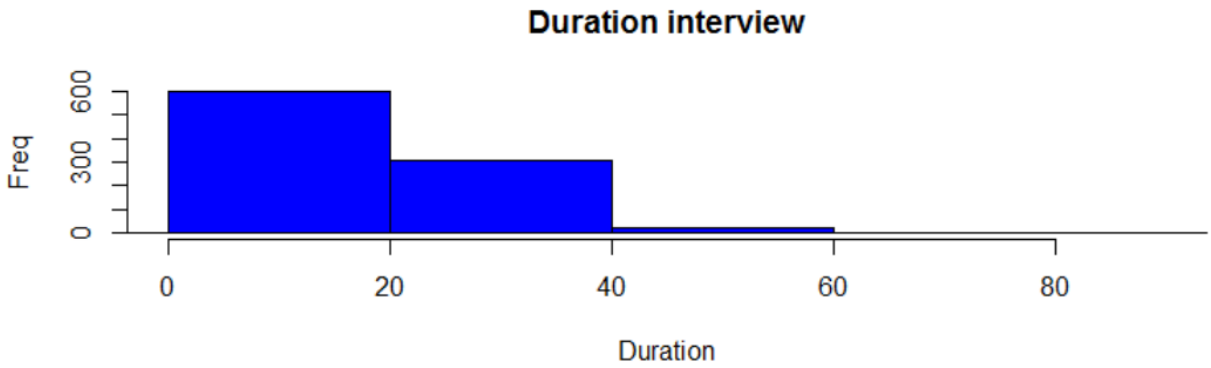


Figure 45:

```
?par
```

The `xlim` argument

```
hist(dat$duration,col="blue",
     main="Duration interview",ylab="Freq", xlab="Duration",
     xlim=c(0,90))
```

The `breaks` argument

- While the previous arguments are valid for many graphics functions, the following applies mainly to histograms:

```
hist(dat$duration,col="red",
     main="Duration of interview", xlab="Duration",
     xlim=c(0,90),breaks=60)
```

- with `breaks` you can control the number of bars...

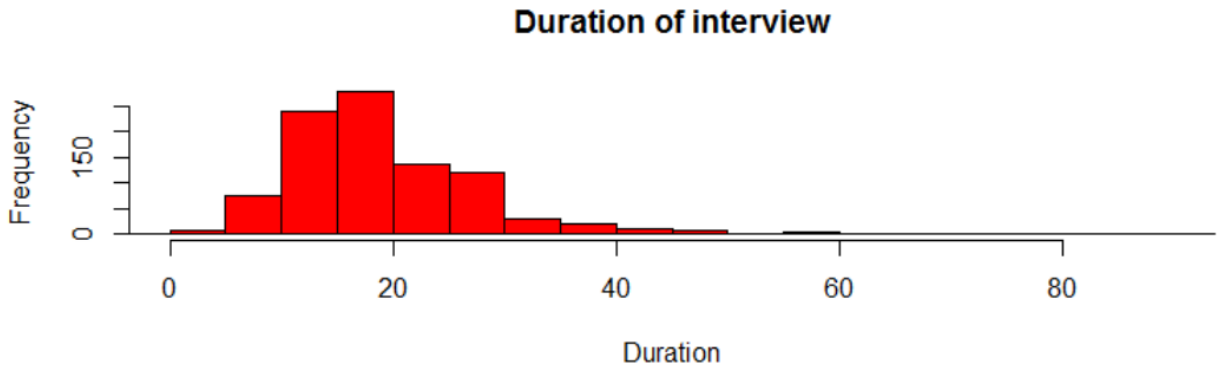


Figure 46:

Tabulate and `barplot`

- The command `barplot()` generates a barplot from a frequency table
- We get the frequency table with the following command:

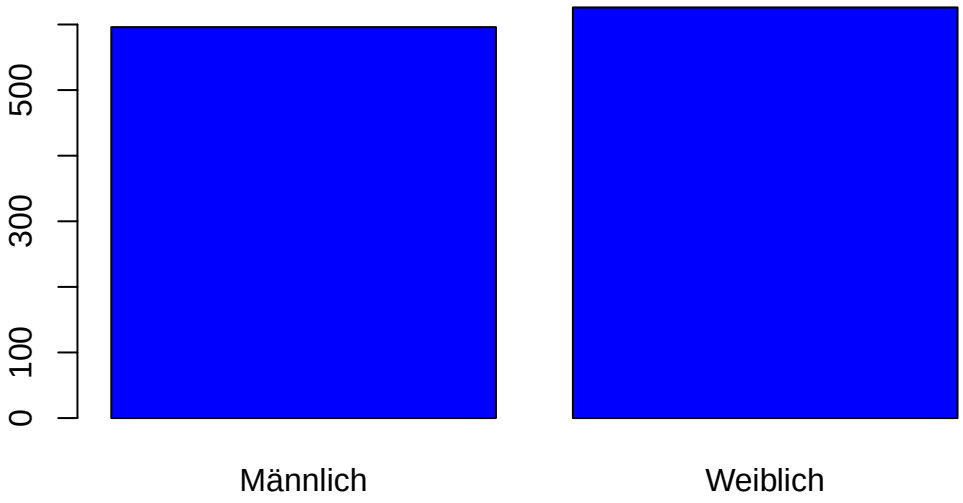
```
tab_sex <- table(dat$a11d054a)
```

```
barplot(tab_sex)
```

- If the passed table object is two-dimensional, a conditional bar plot is created

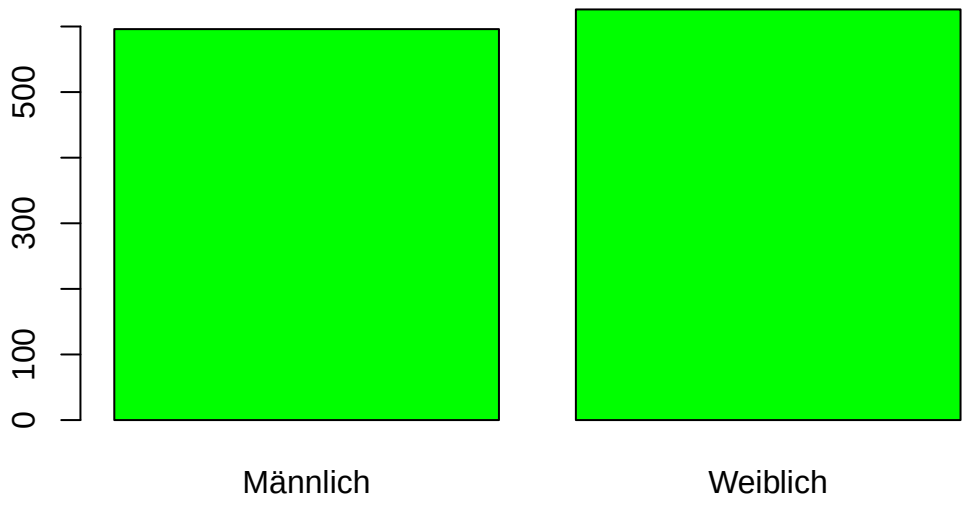
More colour:

```
barplot(tab_sex,col=rgb(0,0,1))
```



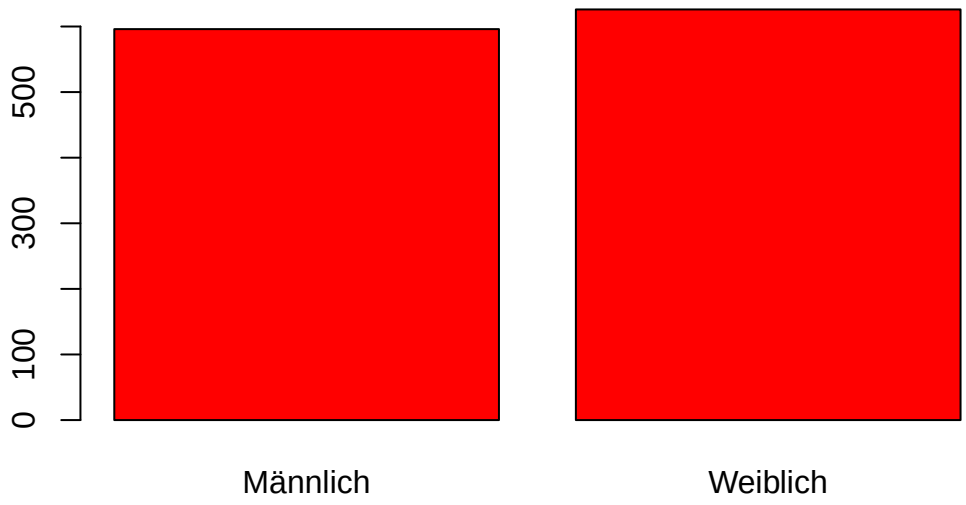
Green colour

```
barplot(tab_sex,col=rgb(0,1,0))
```



Red colour

```
barplot(tab_sex,col=rgb(1,0,0))
```



Transparent

```
barplot(tab_sex,col=rgb(1,0,0,.3))
```

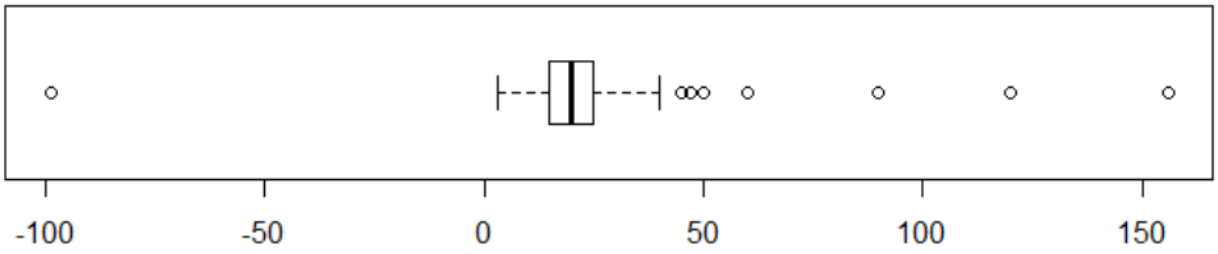
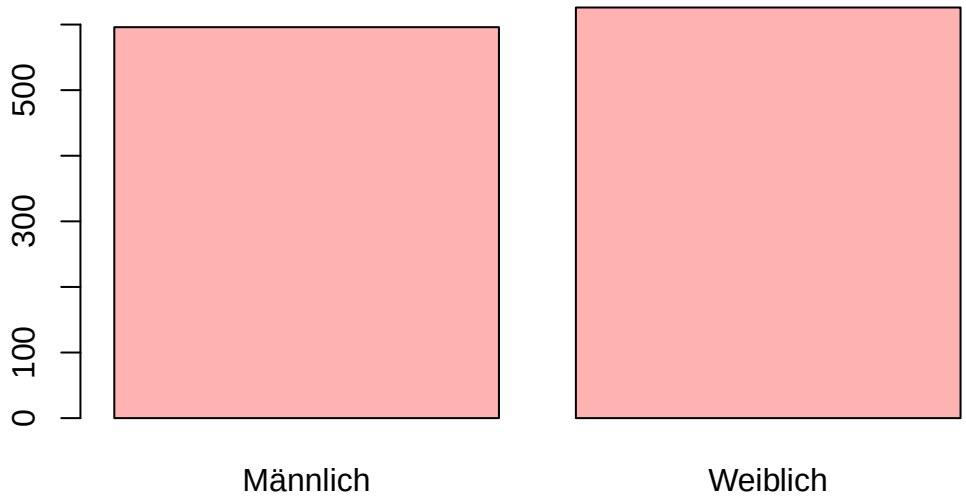


Figure 47:



Horizontal `boxplot`

- A simple **boxplot** can be created with `boxplot()`
- Also `boxplot()` must pass at least one observation vector

```
?boxplot
boxplot(dat$duration,horizontal=TRUE)
```

Grouped boxplots

- A very simple way to get a first impression of conditional distributions is via so-called grouped notched boxplots
- To do this, a so-called formula object must be passed to the `boxplot()` function.
- The conditional variable is located on the right side of a tilde

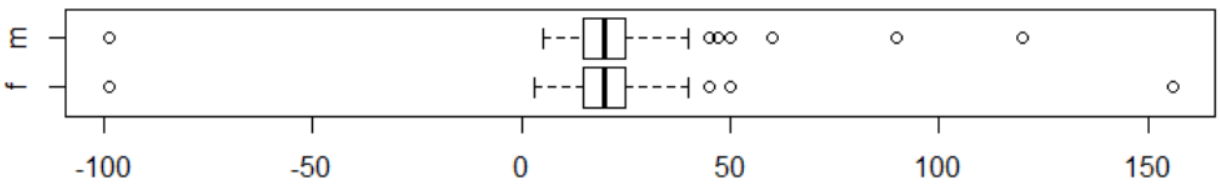


Figure 48:

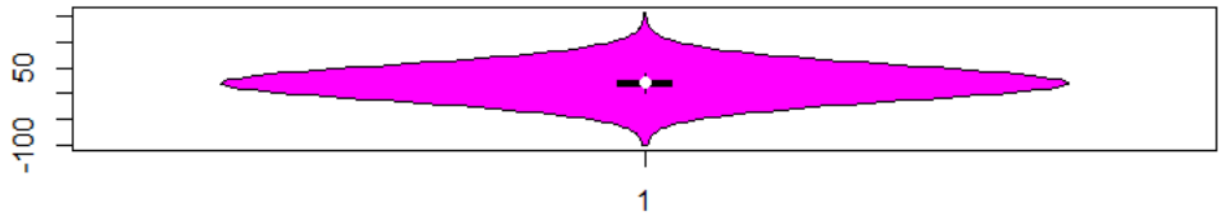


Figure 49:

```
sex <- as.character(dat$a11d054a)
sex[dat$a11d054a=="Männlich"] <- "m"
sex[dat$a11d054a=="Weiblich"] <- "f"
```

```
boxplot(dat$duration~sex,horizontal=TRUE)
```

Boxplot alternatives - `vioplot`

- Builds on Boxplot - additional information about data density
- Density is calculated using the kernel method.
- The further the expansion, the higher the density at this point.
- White dot - median

```
library(vioplot)
vioplot(na.omit(dat$duration))
```

Alternatives `boxplot()`

```
library(beanplot)
par(mfrow = c(1,2))
boxplot(dat$duration~dat$a11d054a,data=dat,col="blue")
beanplot(dat$duration~dat$a11d054a,data=dat,col="orange")
```

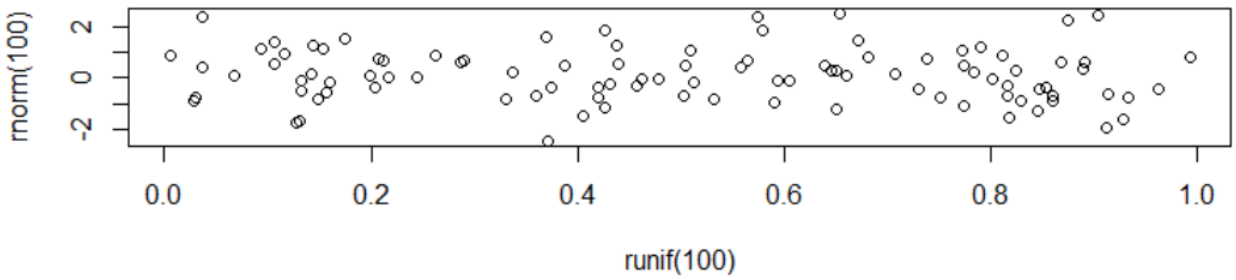
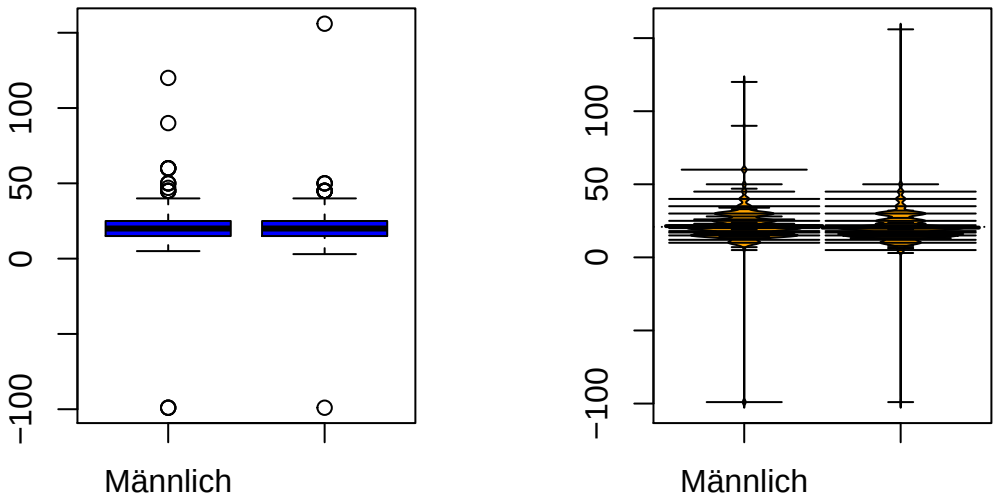


Figure 50:



Conditional, bi- and multivariate distribution graphics - scatterplots

- A simple two-way scatterplot can be created with the `plot()` function
- To create a scatterplot x and y observation vector must be passed
- Argument `col` to specify the color (color as character or numeric)
- Argument `pch` to specify plot symbols (plotting character) (character or numeric)
- The labels are defined with `xlab` and `ylab`.

```
plot(runif(100),rnorm(100))
```

B2A Exercise - simple graphics

- Load the dataset `VADeaths` and create the following plot:

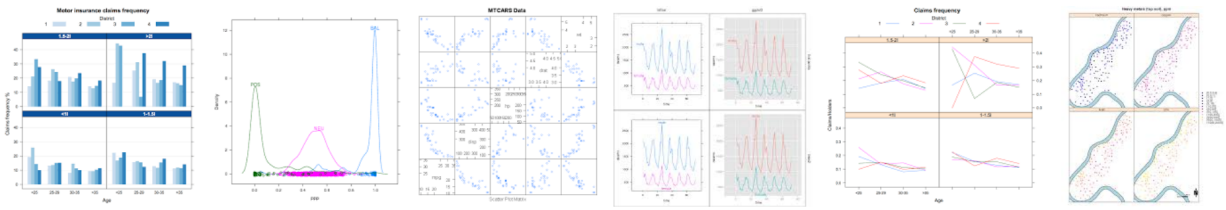
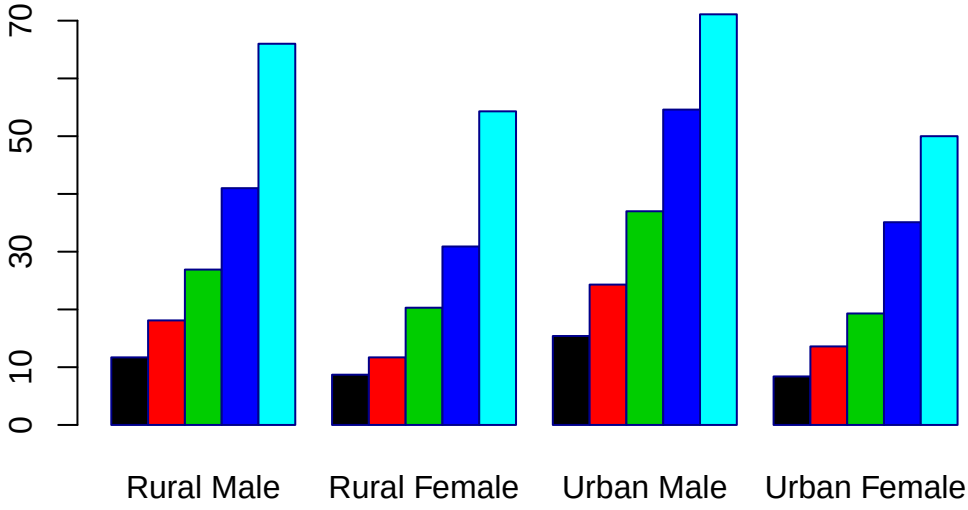


Figure 51:



The `lattice`-Package

Definition of a lattice graphic

> It is designed to meet most typical graphics needs with minimal tuning, but can also be easily extended to handle most nonstandard requirements.

Examples for `lattice` graphics

A addin for RStudio

- install the addinplots package - mark the dataset you want to visualize and choose a plot type:

```
devtools::install_github("homerhanumat/addinplots")
```

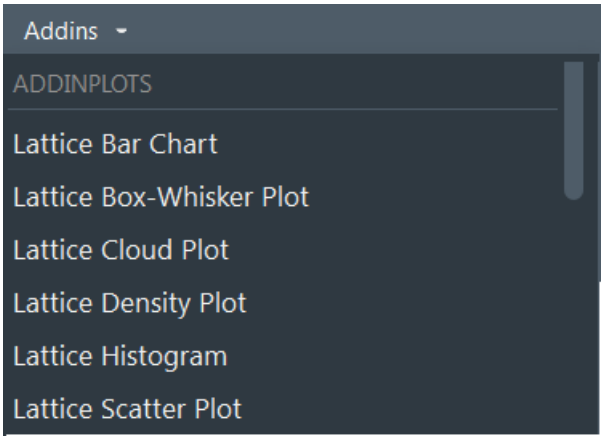


Figure 52:

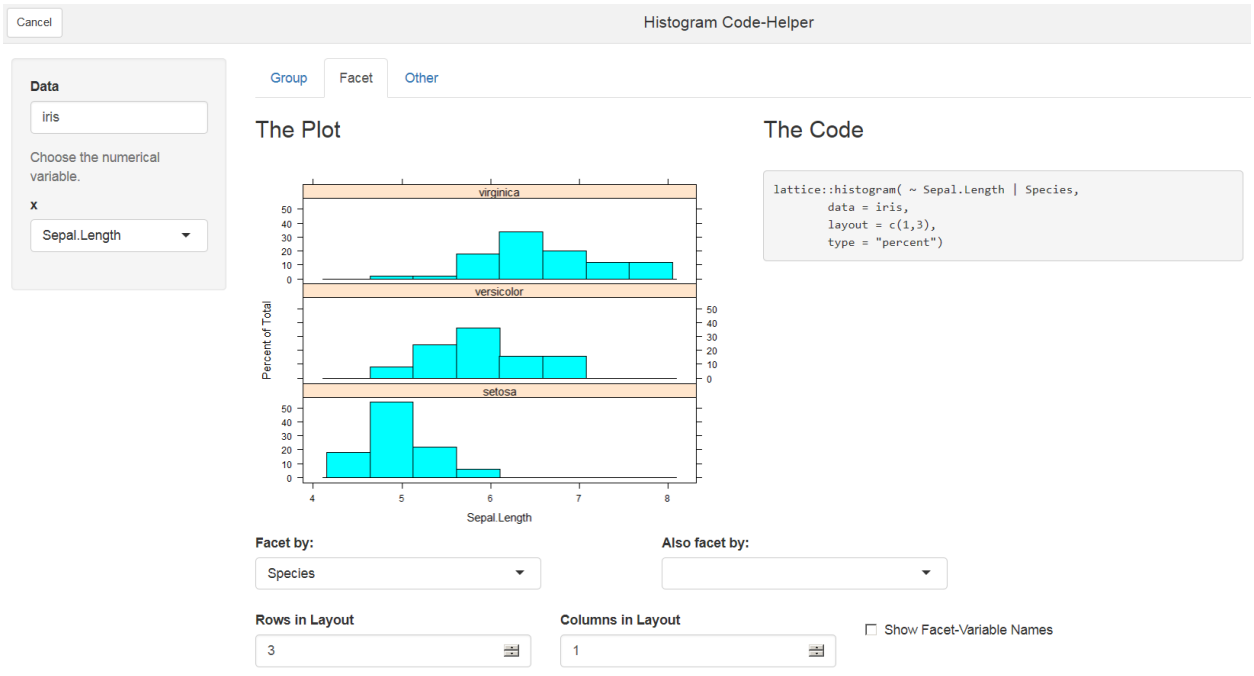


Figure 53:

User interface of `addinplots`

```
iris # example dataset used
```

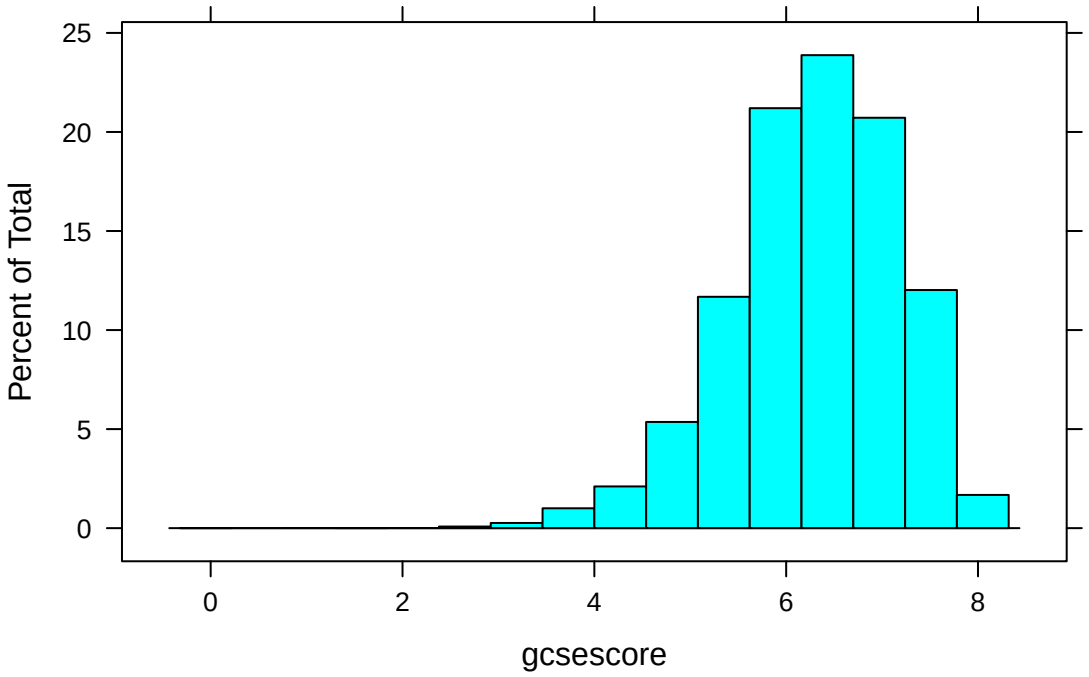
The dataset - Scores on A-level Chemistry in 1997

```
library("mlmRev")
data(Chem97)
```

variables	categories
lea	Local Education Authority
school	School identifier
student	Student identifier
score	Point score on A-level Chemistry in 1997
gender	Student's gender
age	Age in month, centred at 222 months or 18.5 years
gcsescore	Average GCSE score of individual
gcsecnt	Average GCSE score of individual, centered at mean

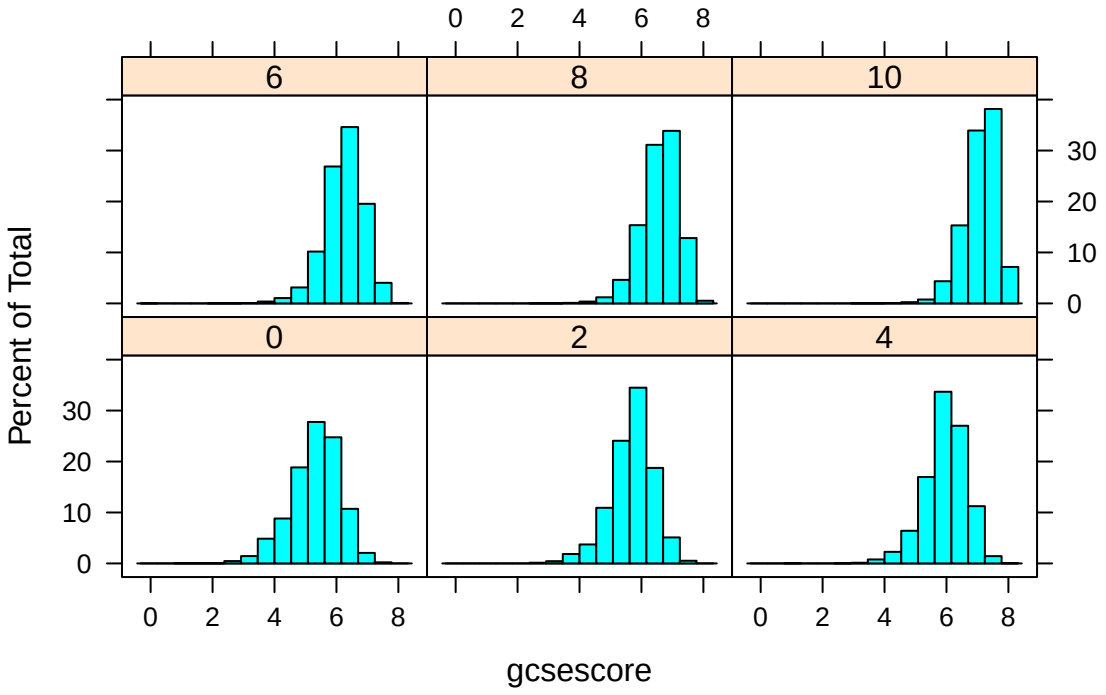
Histogram with `lattice`

```
library("lattice")
histogram(~ gcsescore, data = Chem97)
```



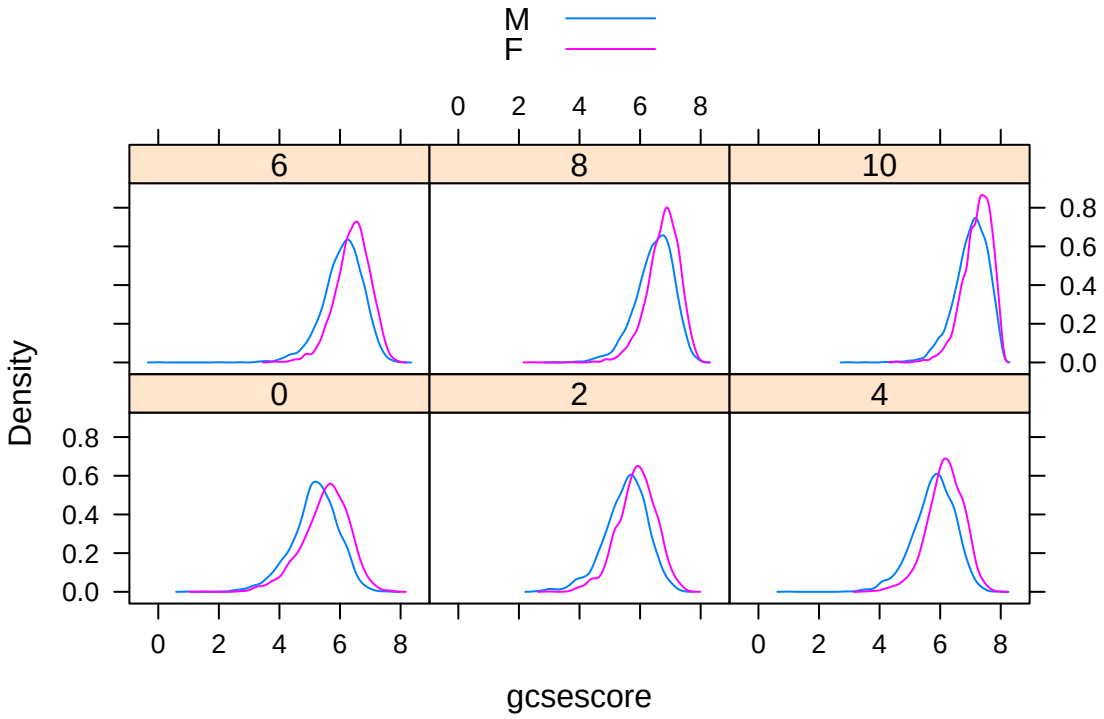
More histograms with `lattice`

```
histogram(~ gcsescore | factor(score),data = Chem97)
```



Plotting the density

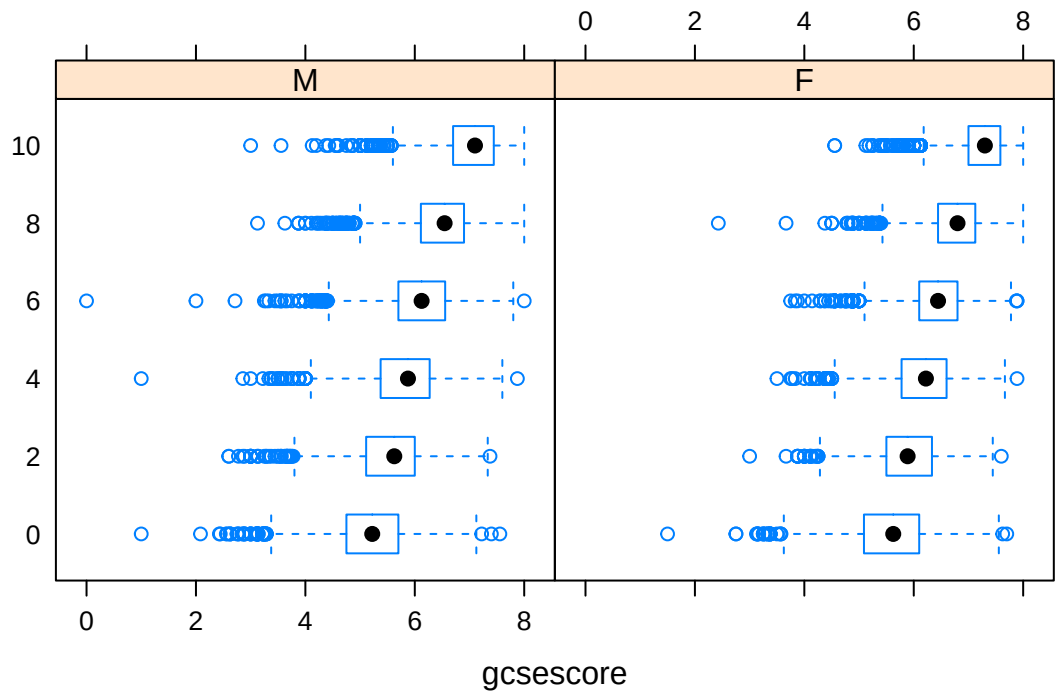
```
densityplot(~ gcsescore | factor(score), Chem97,
    groups=gender,plot.points=FALSE,auto.key=TRUE)
```



Introduction to the `lattice` package

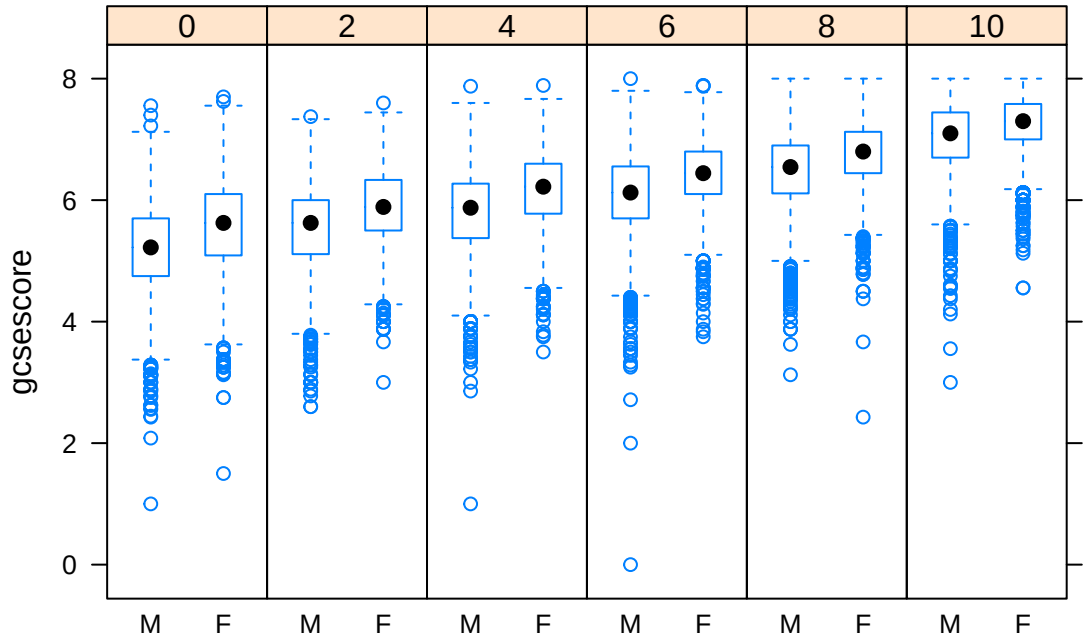
Creating a boxplot with `lattice`

```
bwplot(factor(score) ~ gcsescore | gender, Chem97)
```



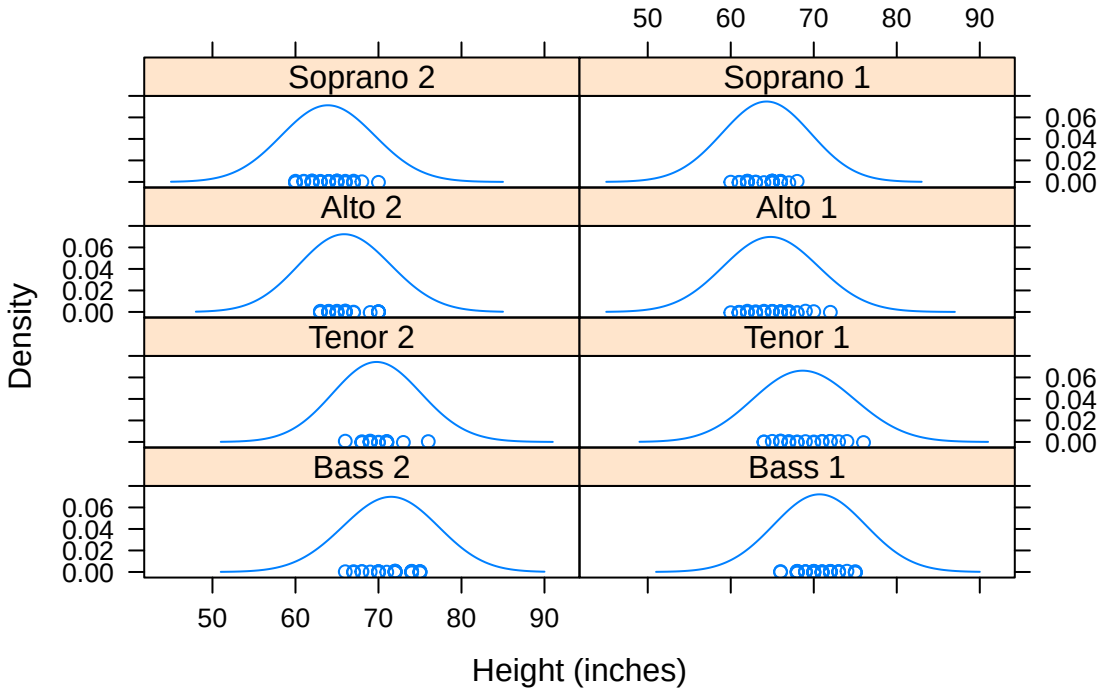
Plotting a boxplot with `lattice`

```
bwplot(gcsescore ~ gender | factor(score), Chem97,
 layout = c(6, 1))
```



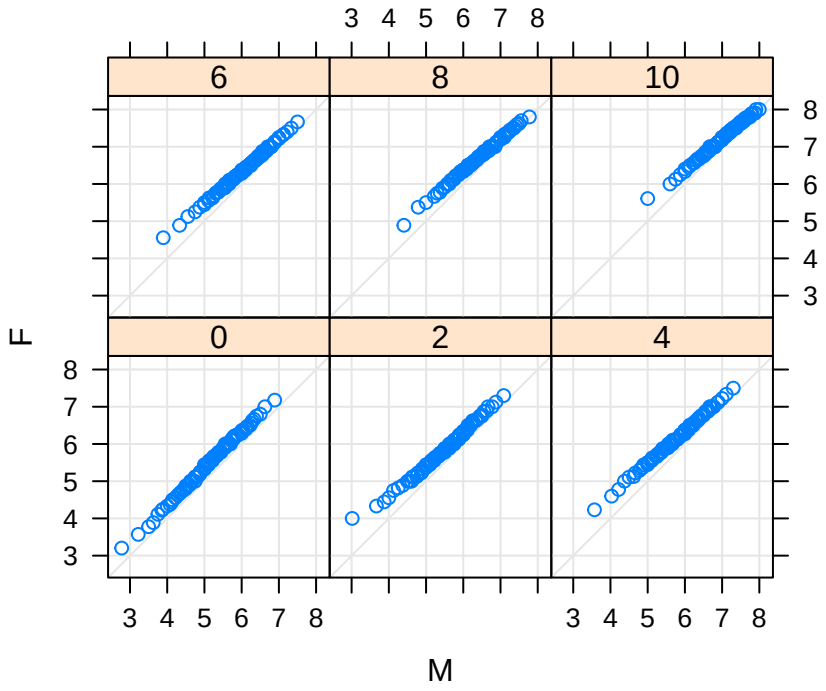
A densityplot

```
densityplot(~height|voice.part,data=singer,layout = c(2,4),
            xlab = "Height (inches)",bw = 5)
```



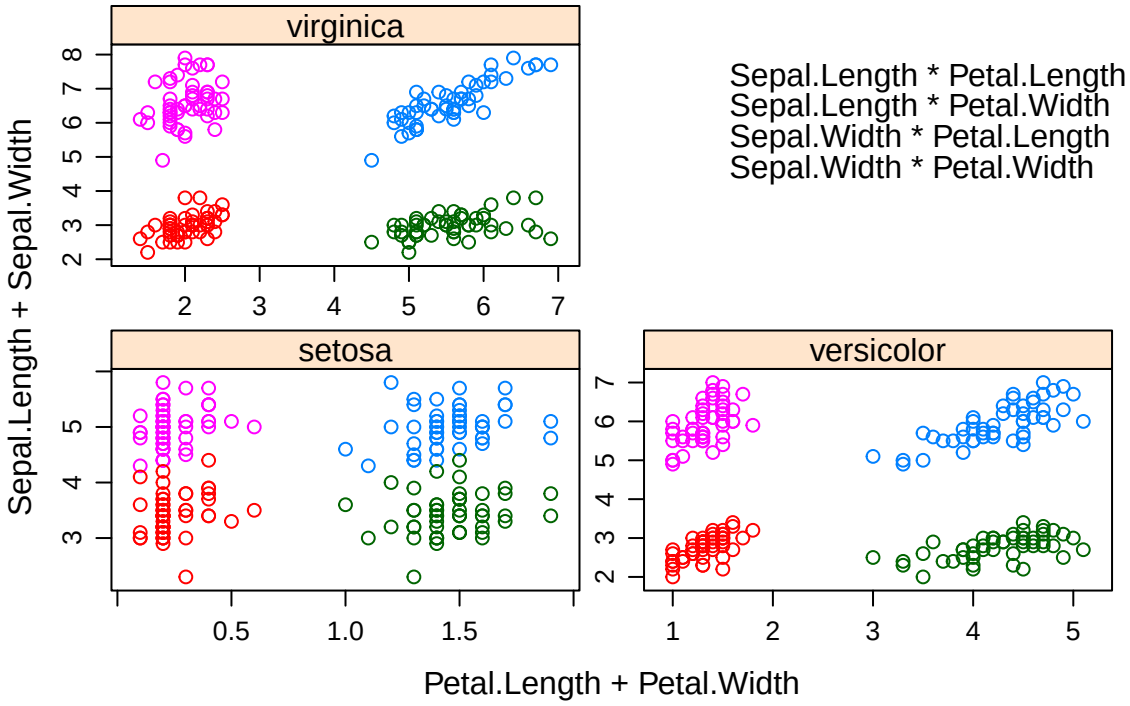
Bivariate Plots

```
qq(gender ~ gcsescore | factor(score), Chem97,
   f.value = ppoints(100), type = c("p", "g"), aspect = 1)
```



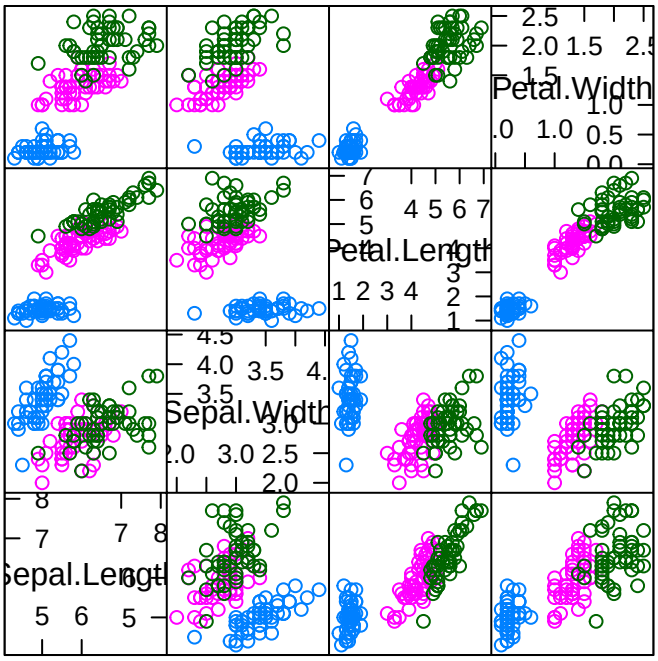
xyplot

```
xyplot(Sepal.Length + Sepal.Width ~ Petal.Length + Petal.Width | Species,
       data = iris, scales = "free", layout = c(2, 2),
       auto.key = list(x = .6, y = .7, corner = c(0, 0)))
```



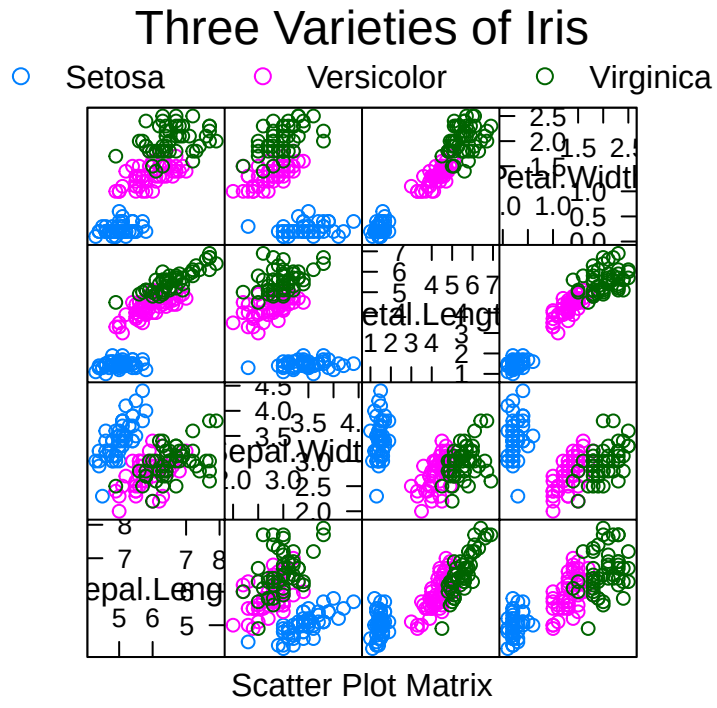
Multivariate plots

```
splom(~iris[1:4], groups = Species, data = iris)
```



Scatter Plot Matrix

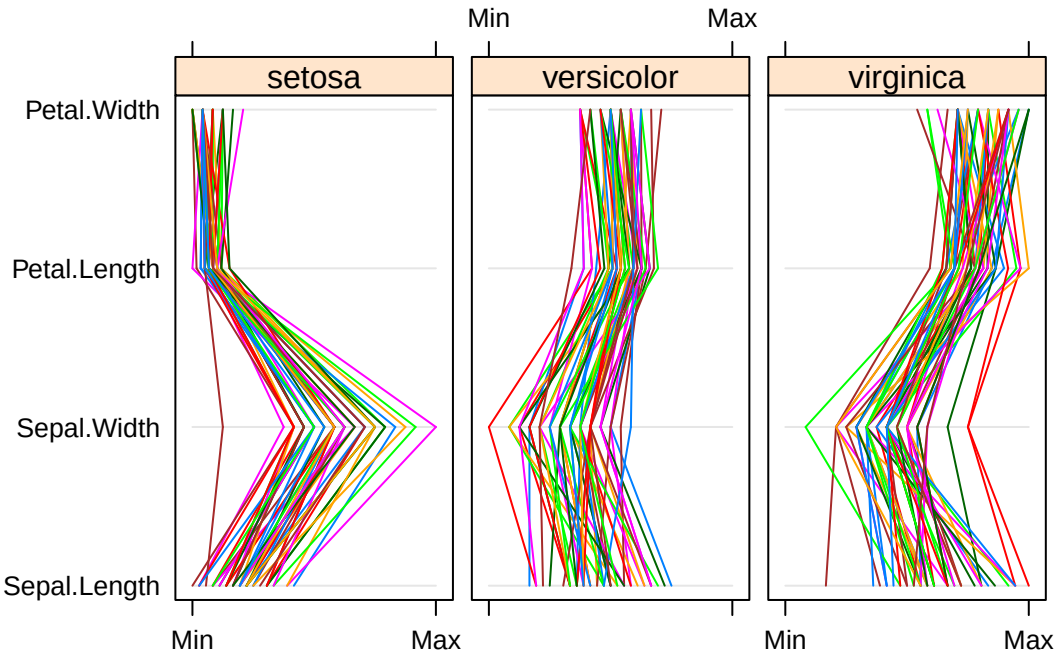
```
super.sym <- trellis.par.get("superpose.symbol")
splom(~iris[1:4], groups = Species, data = iris,
      panel = panel.superpose,
      key = list(title = "Three Varieties of Iris",
                 columns = 3,
                 points = list(pch = super.sym$pch[1:3],
                 col = super.sym$col[1:3]),
                 text = list(c("Setosa", "Versicolor", "Virginica"))))
```



Parallelplot

```
parallelplot(~iris[1:4] | Species, iris)
```

Figure 54:



The ggplot2 package

Introduction ggplot2

The ggplot2 package, created by Hadley Wickham, offers a powerful graphics language for creating elegant and complex plots. Its popularity in the R community has exploded in recent years. Origianlly based on Leland Wilkinson's The Grammar of Graphics, ggplot2 allows you to create graphs that represent both univariate and multivariate numerical and categorical data in a straightforward manner.

Examples ggplot2 graphics

A first example ggplot2

```
library(ggplot2)
ggplot(midwest, aes(x=area, y=poptotal)) + geom_point()
```

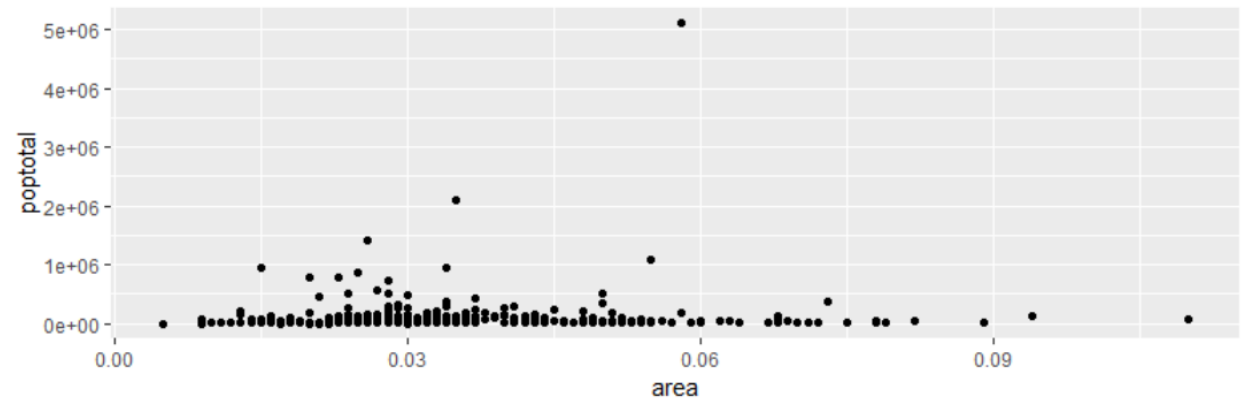


Figure 55:

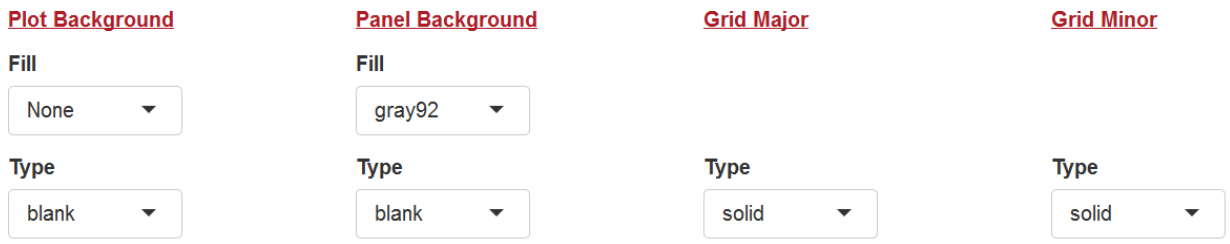


Figure 56:

Some nice Rstudio Addins

- A `ggplot` graphic has to be marked in source code, to use the following addins

```
install.packages("ggThemeAssist")
```

```
install.packages('ggedit')
```

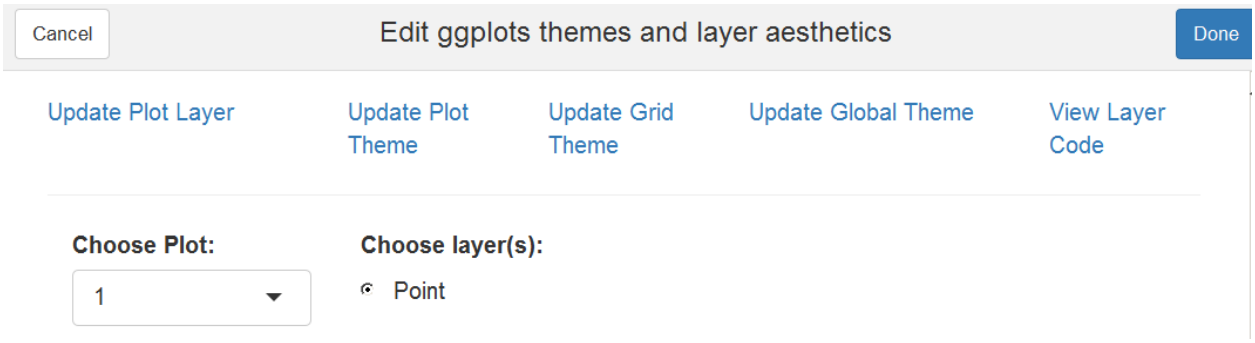


Figure 57:

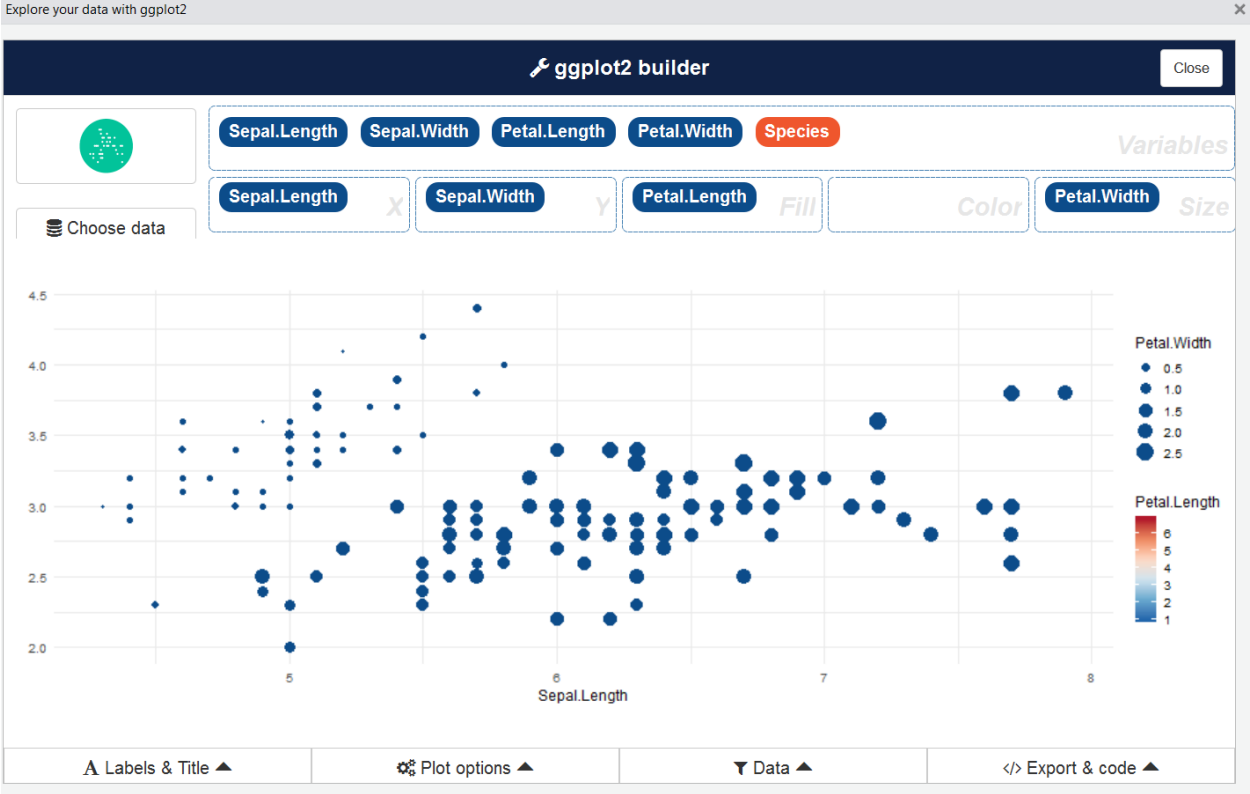


Figure 58:

ggplot2 builder addin for RStudio

```
devtools::install_github("dreamRs/esquisse")
```

Links

- J H Maindonald - Lattice and Other Graphics in R
- DEEPAYAN SARKAR - AN INTRODUCTION TO R - lattice lab
- FLOWINGDATA - Comparing ggplot2 and R Base Graphics
- Quick R - ggplot2
- Top 50 ggplot2 Visualizations
- Bioconductor R manual with an extensive part on graphics

Linear regression

Good literature for linear regression in R

Useful PDF document:

J H Maindonald - Using R for Data Analysis and Graphics Introduction, Code and Commentary

- Introduction to R
- Data analysis
- Statistical models
- Inference concepts
- Regression with one predictor
- Multiple linear regression
- Extending the linear model
- ...

Variables of the `mtcars` dataset

Help for the `roller` dataset:

```
?mtcars
```

- mpg - Miles/(US) gallon
- cyl - Number of cylinders
- disp - Displacement (cu.in.)
- hp - Gross horsepower
- drat - Rear axle ratio
- wt - Weight (1000 lbs)
- qsec - 1/4 mile time
- vs - Engine (0 = V-shaped, 1 = straight)
- am - Transmission (0 = automatic, 1 = manual)
- gear - Number of forward gears
- carb - Number of carburetors

Dataset `mtcars`

	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
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	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 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3

	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
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

A simple regression model

Dependent variable - miles per gallon (mpg)

Independent variable - weight (wt)

```
m1 <- lm(mpg ~ wt,data=mtcars)
m1
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

Get the model summary

```
summary(m1)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

The model formula

Model without intercept

```
m2 <- lm(mpg ~ - 1 + wt,data=mtcars)
summary(m2)$coefficients
```

```
##    Estimate Std. Error  t value    Pr(>|t|)
## wt 5.291624  0.5931801 8.920771 4.55314e-10
```

Adding further variables

```
m3 <- lm(mpg ~ wt + cyl,data=mtcars)
summary(m3)$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 39.686261  1.7149840 23.140893 3.043182e-20
## wt          -3.190972  0.7569065 -4.215808 2.220200e-04
## cyl         -1.507795  0.4146883 -3.635972 1.064282e-03
```

Further possibilities to specify the formula

Interaction effect

```
# effect of cyl and interaction effect:
m3a<-lm(mpg~wt*cyl,data=mtcars)

# only interaction effect:
m3b<-lm(mpg~wt:cyl,data=mtcars)
```

Take the logarithm

```
m3d<-lm(mpg~log(wt),data=mtcars)
```

Exploring interactions

```
install.packages("jtools")
```

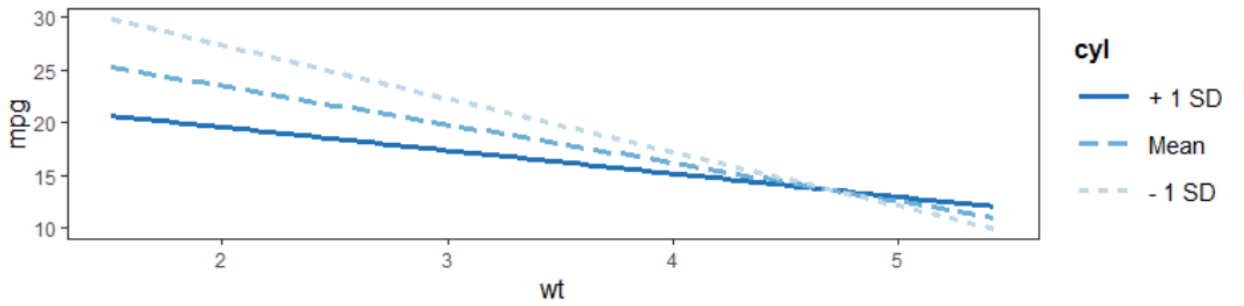


Figure 59:

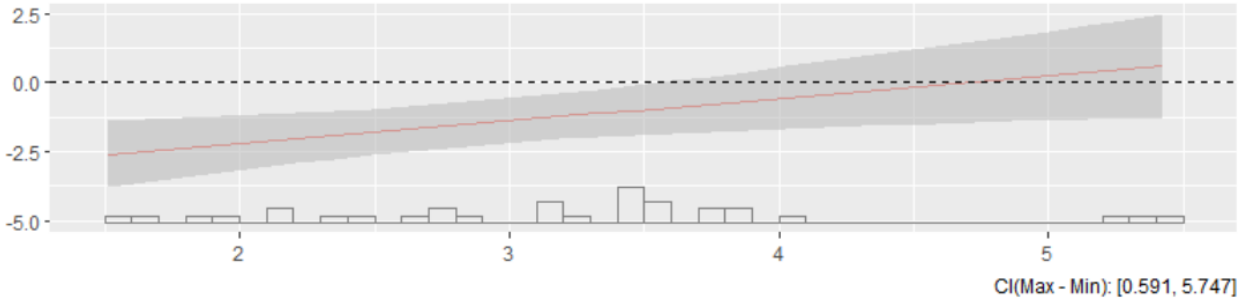


Figure 60:

```
library(jtools)
interact_plot(m3a, pred = "wt", modx = "cyl")
```

The package `interplot`

- `interplot` vignette

```
library(interplot)
```

```
interplot(m = m3a, var1 = "cyl", var2 = "wt", hist = TRUE) +
  aes(color = "pink") + theme(legend.position="none") +
  geom_hline(yintercept = 0, linetype = "dashed")
```

Example: object orientation

- m3 is now a special regression object
- Various functions can now be applied to this object

```
predict(m3) # Prediction
resid(m3) # Residuals
```

```
##         Mazda RX4     Mazda RX4 Wag        Datsun 710    Hornet 4 Drive
##          22.27914          21.46545          26.25203          20.38052
## Hornet Sportabout           Valiant
##          16.64696          19.59873
```

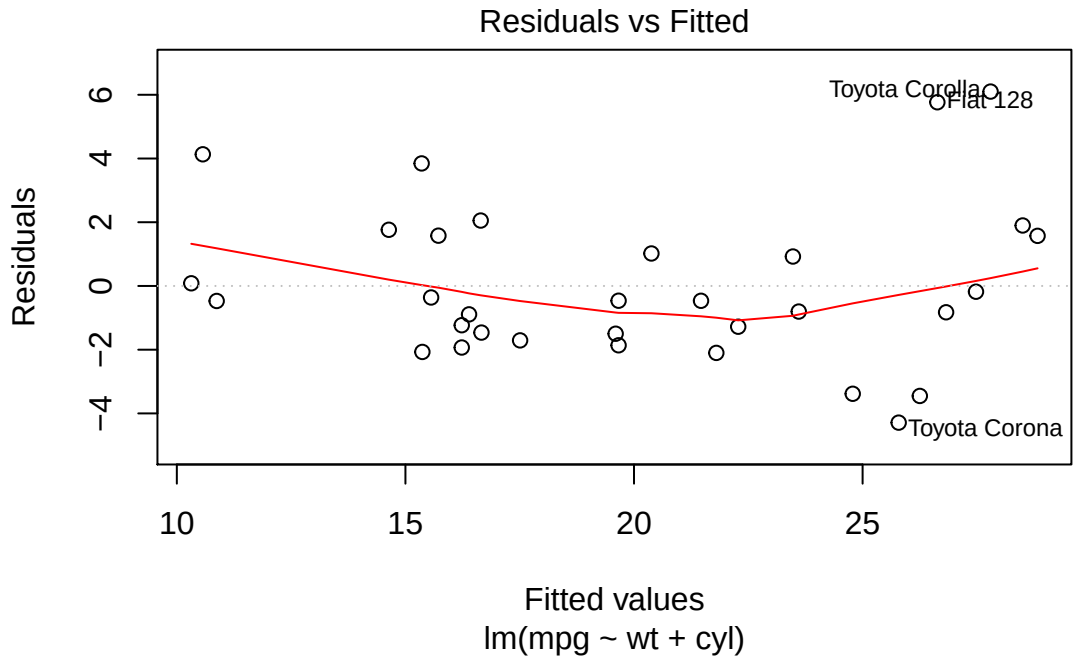
```
##         Mazda RX4     Mazda RX4 Wag        Datsun 710    Hornet 4 Drive
```

```
##        -1.2791447        -0.4654468        -3.4520262         1.0194838
## Hornet Sportabout             Valiant
##         2.0530424        -1.4987281
```

Residual plot - model assumptions violated?

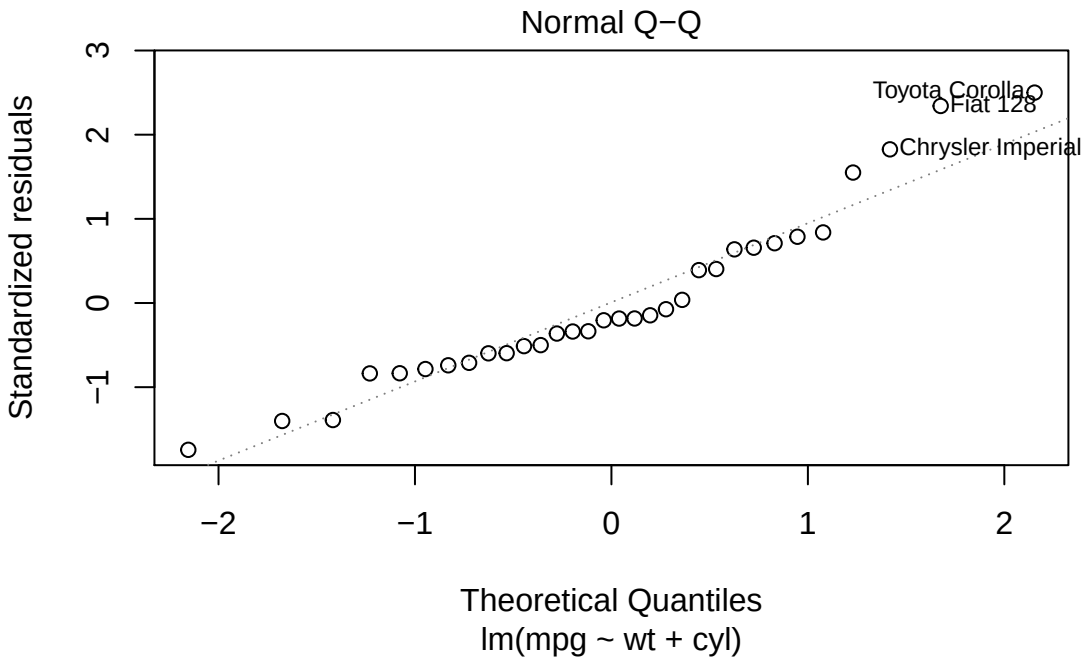
- the case if a deviation pattern from line

```
plot(m3,1)
```



Residual plot

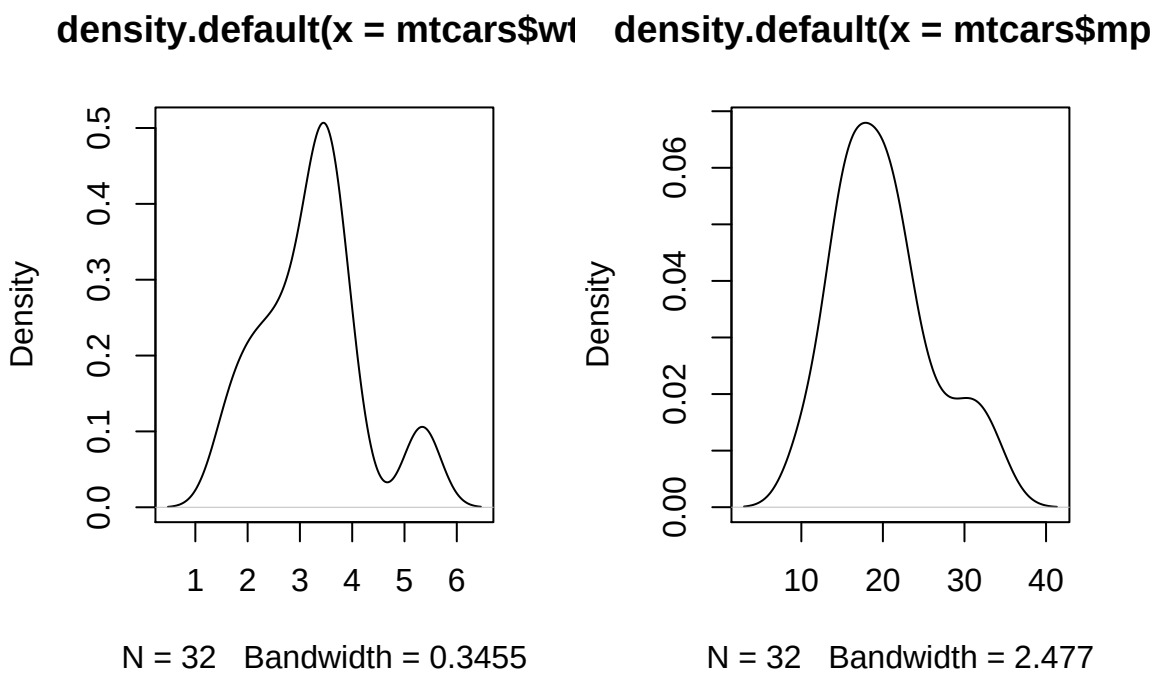
```
plot(m3,2)
```



- If the residuals are normally distributed, they should be on the same line.

A simple regression model

```
par(mfrow=c(1,2))
plot(density(mtcars$wt)); plot(density(mtcars$mpg))
```



Make model prediction

```
pre <- predict(m1)
head(mtcars$mpg)
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1
```

```
head(pre)
```

```
##         Mazda RX4     Mazda RX4 Wag        Datsun 710    Hornet 4 Drive
##          23.28261          21.91977          24.88595          20.10265
## Hornet Sportabout           Valiant
##          18.90014          18.79325
```

Regression diagnostic with base-R

```
plot(mtcars$wt,mtcars$mpg)
abline(m1)
segments(mtcars$wt, mtcars$mpg, mtcars$wt, pre, col="red")
```

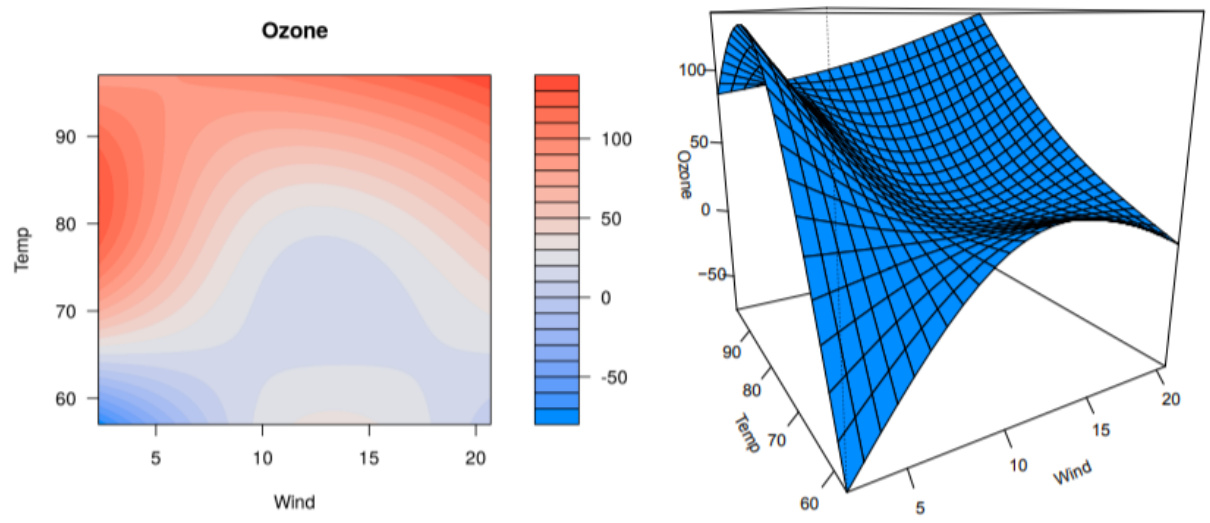
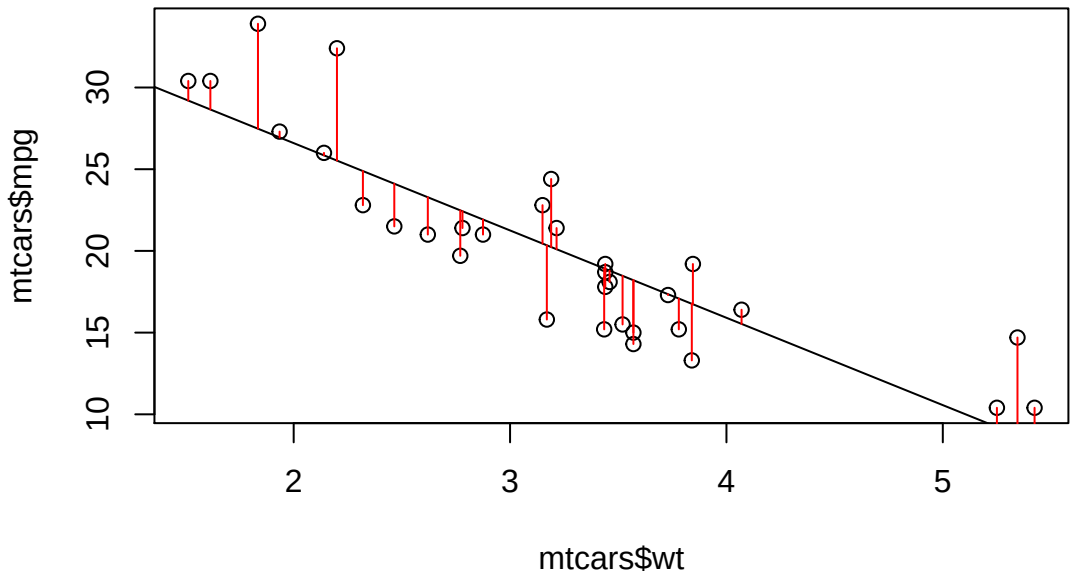


Figure 61:



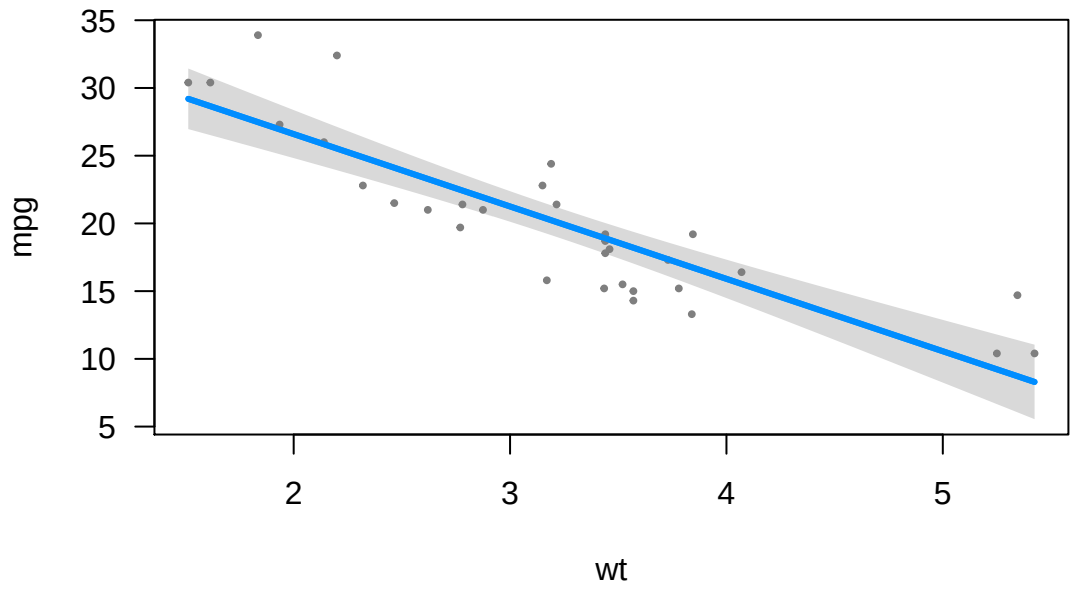
The `visreg`-package

```
install.packages("visreg")
```

```
library(visreg)
```

Visualisation

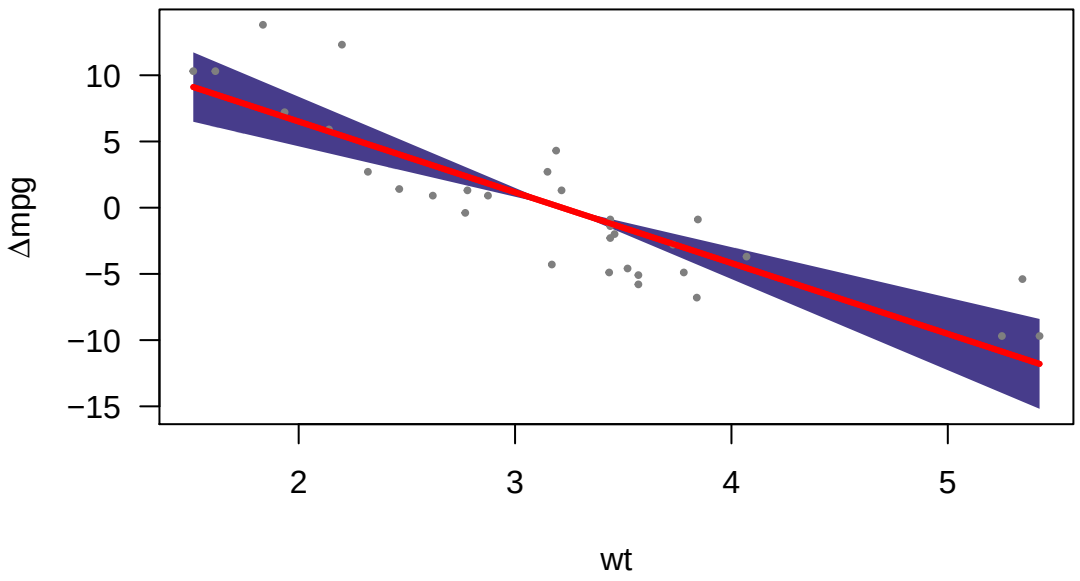
```
visreg(m1)
```



Visualisation with `visreg`

- Second argument - Specification covariate for visualisation

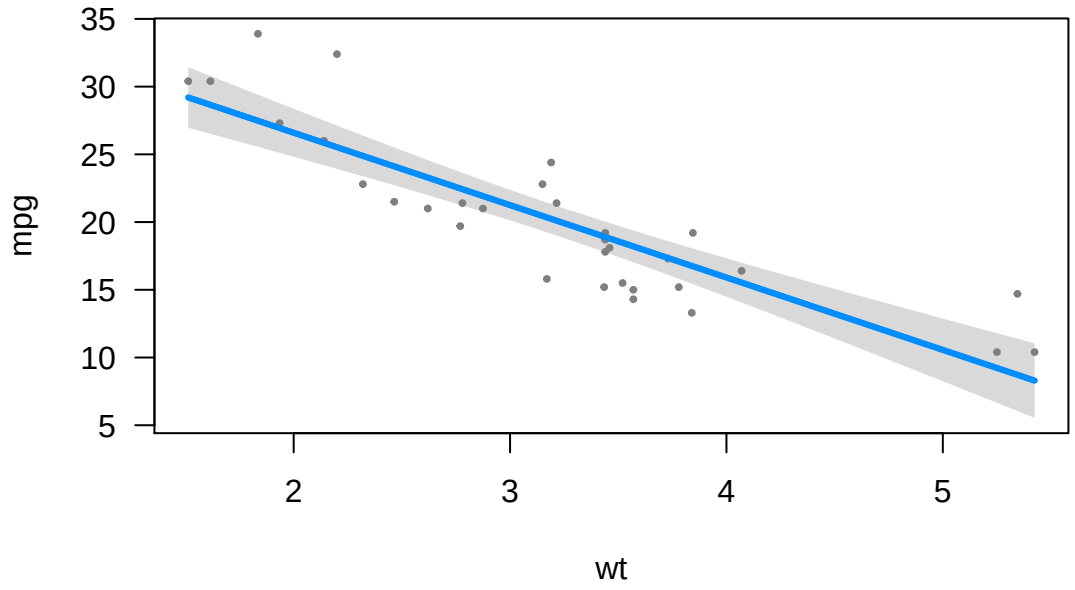
```
visreg(m1, "wt", type = "contrast")
```



The `visreg`-package

- The default-argument for `type` is `conditional`.

```
visreg(m1, "wt", type = "conditional")
```



Regression with factors

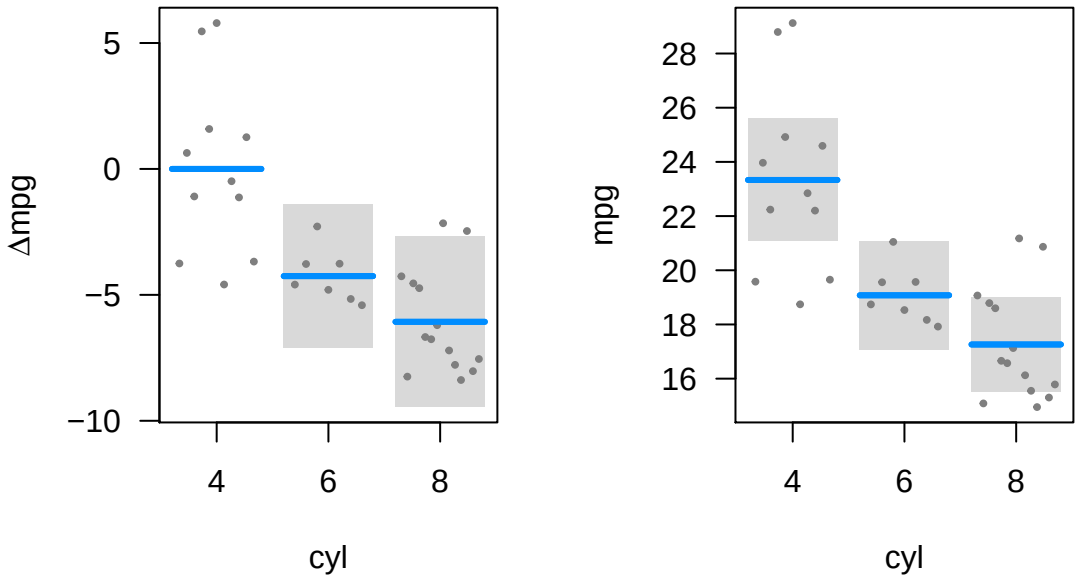
- With `visreg` the effects of factors can be visualized.

```
mtcars$cyl <- as.factor(mtcars$cyl)
m4 <- lm(mpg ~ cyl + wt,
    data = mtcars)
summary(m4)
```

```
##
## Call:
## lm(formula = mpg ~ cyl + wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5890 -1.2357 -0.5159  1.3845  5.7915
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  33.9908     1.8878  18.006  < 2e-16 ***
## cyl6         -4.2556     1.3861  -3.070 0.004718 **
## cyl8         -6.0709     1.6523  -3.674 0.000999 ***
## wt           -3.2056     0.7539  -4.252 0.000213 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.557 on 28 degrees of freedom
## Multiple R-squared:  0.8374, Adjusted R-squared:   0.82
## F-statistic: 48.08 on 3 and 28 DF,  p-value: 3.594e-11
```

Effects of factors

```
par(mfrow=c(1,2))
visreg(m4, "cyl", type = "contrast")
visreg(m4, "cyl", type = "conditional")
```



The package visreg - Interactions

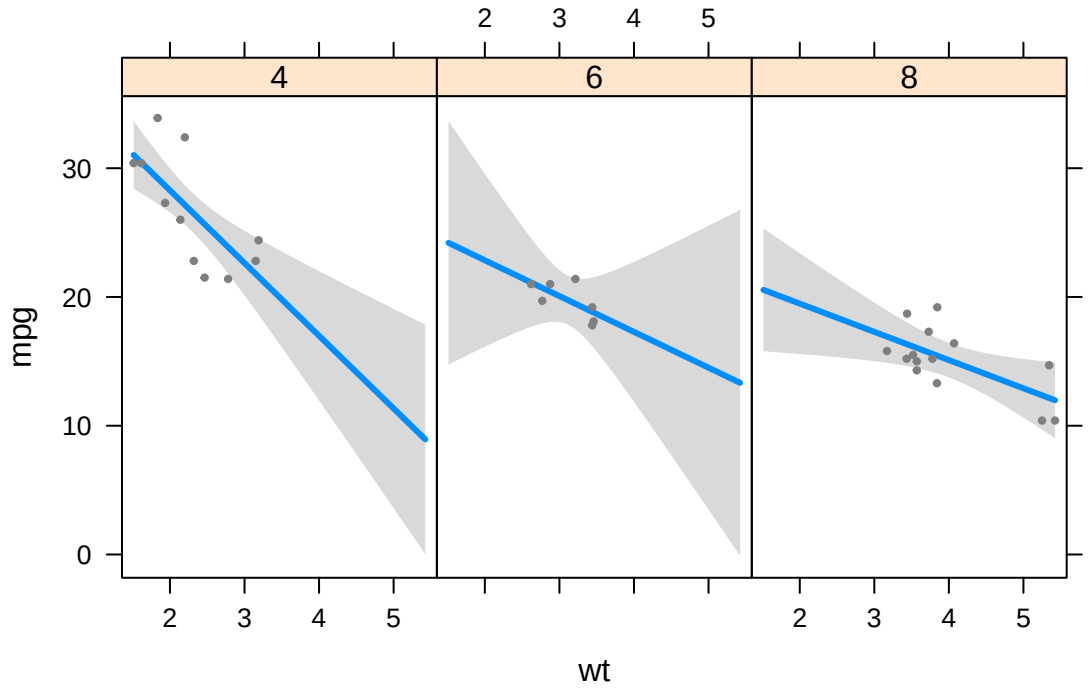
```
m5 <- lm(mpg ~ cyl*wt, data = mtcars)
summary(m5)
```

```
##
## Call:
## lm(formula = mpg ~ cyl * wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1513 -1.3798 -0.6389  1.4938  5.2523
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   39.571      3.194  12.389 2.06e-12 ***
## cyl6         -11.162      9.355  -1.193 0.243584
## cyl8         -15.703      4.839  -3.245 0.003223 **
## wt            -5.647      1.359  -4.154 0.000313 ***
## cyl6:wt        2.867      3.117   0.920 0.366199
## cyl8:wt        3.455      1.627   2.123 0.043440 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.449 on 26 degrees of freedom
## Multiple R-squared:  0.8616, Adjusted R-squared:  0.8349
## F-statistic: 32.36 on 5 and 26 DF,  p-value: 2.258e-10
```

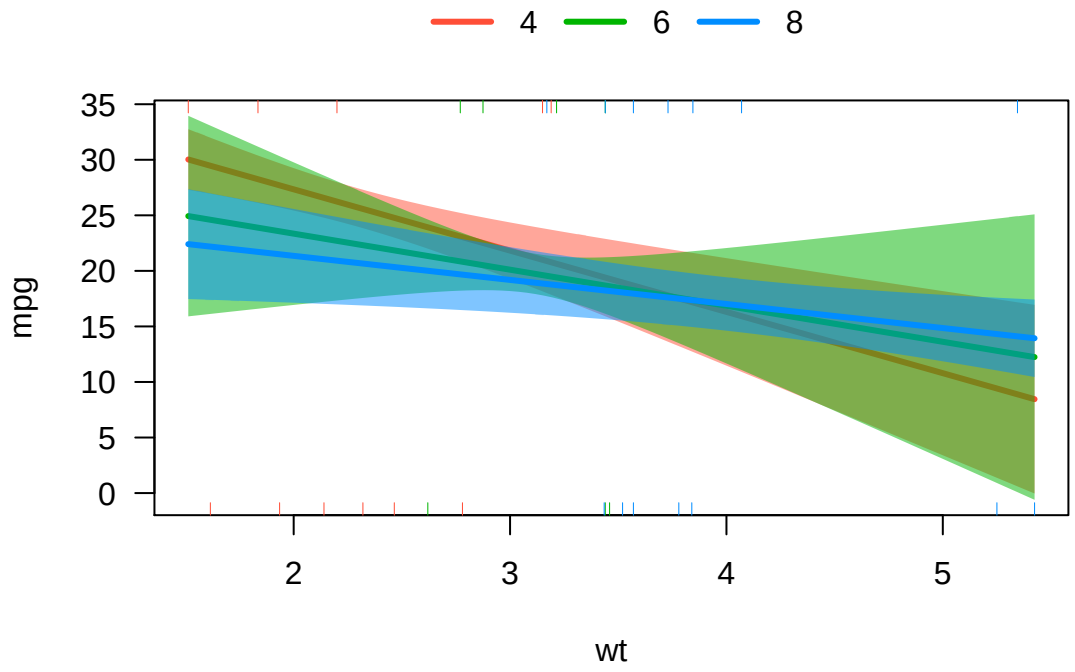
Control of the graphic output with `layout`.

```
visreg(m5, "wt", by = "cyl",layout=c(3,1))
```



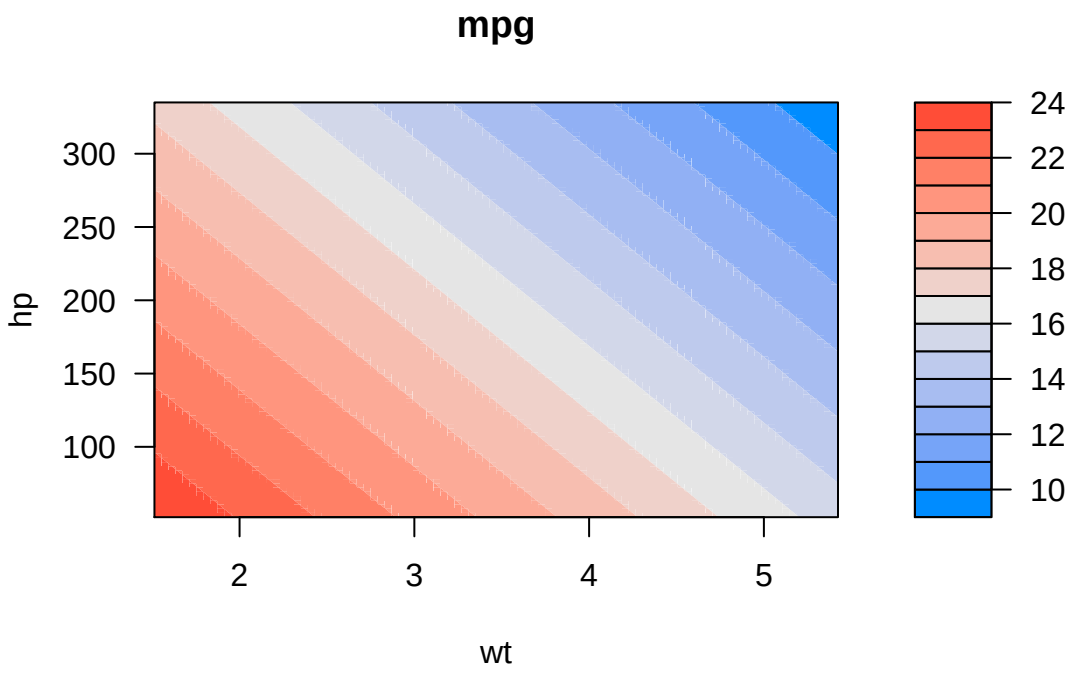
The package `visreg` - Interactions overlay

```
m6 <- lm(mpg ~ hp + wt * cyl, data = mtcars)
visreg(m6, "wt", by="cyl", overlay=TRUE, partial=FALSE)
```



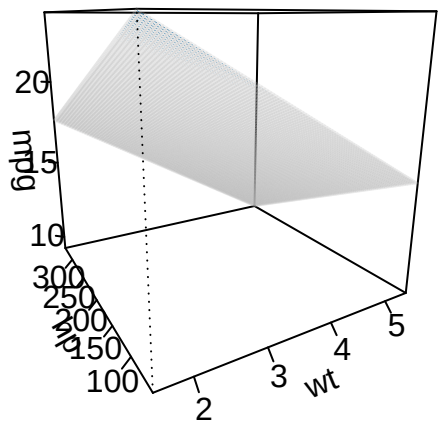
The package `visreg` - `visreg2d`

```
visreg2d(m6, "wt", "hp", plot.type = "image")
```



The package `visreg` - `surface`

```
visreg2d(m6, "wt", "hp", plot.type = "persp")
```



B3A Exercise linear regression

It describes the route of three toy cars that descended ramps at different angles.

- angle: ramp angle
- distance: distance covered by the toy car
- car: type of car (1, 2 or 3)

(a) Read the data set `toycars` into a dataframe `data` and convert the variable `car` of the data set into a factor (`as.factor`).

(b) Create three box plots that display the distance traveled separately by the factor car.

(c) For the cars, estimate the parameters of the following linear model using the `lm()` function

$$distance_i = \beta_0 + \beta_1 \cdot angle_i + \epsilon_i$$

(d) Check the adjustment of the three models by inserting the regression line into a plot of `distance` against `angle`. Does the

$$R^2$$

each for a good model adjustment?

Producing nice table output with Package stargazer

```
library(stargazer)
stargazer(m3, type="html")
```

Example HTML output:

Links - linear regression

- Regression - r-bloggers
- The complete book of Faraway- very intuitive
- Good introduction on Quick-R
- Multiple regression
- 15 Types of Regression you should know
- `ggeffects` - Create Tidy Data Frames of Marginal Effects for 'ggplot' from Model Outputs

Logistic regression

Agresti - Categorical Data Analysis (2002)

- Very intuitively written book
- Very detailed accompanying script by **Laura A. Thompson**
- The paper deals with categorical data analysis in general.

Faraway books on regression with R

- Logistic regression intuitively explained
- Examples with R-code
  - Faraway - **Extending the linear model with R**
  - Faraway - **Practical Regression and Anova using R**

A function to recode the missing values

```
code_miss <- function(var){
  misvals <- c(-11,-22,-33,-44,-55,-66,-77,-88,-99,-111)
  var[var %in% misvals] <- NA
  return(var)
}
```

Variables for `glm`'s

- a11d056z: age group

```
table(datf$a11d056z)
```

```
##
## -99   1   2   3   4   5   6   7   8   9  10  11  12  13
##   5  31  87 101  91  83 100 163 159 133  64  56 105  44
```

```
age <- code_miss(datf$a11d056z)
```

```
table(age)
```

	Dependent variable:
	mpg
wt	-3.125***
	(0.911)
cyl	-1.510***
	(0.422)
am	0.176
	(1.304)
Constant	39.418***
	(2.641)
Observations	32
R^2	0.830
Adjusted R^2	0.812
Residual Std. Error	2.612 (df = 28)
F Statistic	45.678*** (df = 3; 28)
Note:	*p<0.1; **p<0.05; ***p<0.01

Figure 62:

Figure 63:

Figure 64:

```
## age
##   1   2   3   4   5   6   7   8   9  10  11  12  13
##  31  87 101  91  83 100 163 159 133  64  56 105  44
```

GP variable a11d094a: Children under 16 years

Does your household include children under 16?

- 1 Yes
- 2 No

```
children <- as.factor(code_miss(datf$a11d094a))
table(children)
```

```
## children
##   1   2
## 325 681
```

Conditional Density Plot (GESIS Panel)

```
cdplot(children ~ age, data = dat)
```

Binary independent variables with `glm`

- The logistic regression belongs to the class of generalized linear models (GLM)
- The function for estimating a model of this class in is called `glm()`
- `glm()`

Specifying a `glm`

- formula object
- the class (binomial, gaussian, gamma)
- including link function (logit, probit, cauchit, log, cloglog)

must be specified

Logistic regression with R

```
glm_1 <- glm(children ~ age,
                    family = binomial())
```

```
sum_glm1 <- summary(glm_1)
sum_glm1$coefficients
```

```
##               Estimate Std. Error   z value     Pr(>|z|)
## (Intercept) -0.7194058 0.16384386 -4.390801 1.129338e-05
## age          0.2225862 0.02376266  9.367056 7.458415e-21
```

llh	The log-likelihood from the fitted model
llhNull	The log-likelihood from the intercept-only restricted model
G2	Minus two times the difference in the log-likelihoods
McFadden	McFadden's pseudo r-squared
r2ML	Maximum likelihood pseudo r-squared
r2CU	Cragg and Uhler's pseudo r-squared

Figure 65:

Interpreting the results

```
anova(glm_1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: children
##
## Terms added sequentially (first to last)
##
##
##      Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
## NULL                  1000       1259
## age   1   98.956       999       1160 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Mc Fadden's R^2

```
library(pscl)
pR2(glm_1)
```

```
##           llh       llhNull            G2      McFadden          r2ML
## -580.02210772 -632.93066002  105.81710461    0.08359297    0.10031573
##          r2CU
##    0.13978426
```

Distance between residential area and large city

How far is it from where you live to the center of the nearest large city?

- 1 - In the center of a big city
- 6 - 60 km and more

```
region <- code_miss(datf$bczd001a)
table(region)
```

```
## region
##   1   2   3   4   5   6
```

```
##  87 191 279 157 126 165
```

Satisfaction life in place of residence

How satisfied are you - all in all - with your life in [place of residence] at the moment?

- 1 - Very satisfied
- 5 - Very dissatisfied

```
satisfactionplace <- datf$a11c019a
table(satisfactionplace)
```

```
## satisfactionplace
##   1   2   3   4   5
## 553 534  99  30   6
```

Another model

```
glm_2 <- glm(children ~ age + satisfactionplace*region,
                    family = binomial())
```

```
pseudor2 <- pR2(glm_2)
pseudor2["McFadden"]
```

```
## McFadden
## 0.258121
```

Another variable in the Gesis Panel data

- Number of tattoos:

```
Tatoos <- code_miss(datf$bdao067a)
Tatoos[Tatoos==97]<-0
```

```
table(Tatoos)
```

```
## Tatoos
##   0   1   2   3   4   5   6
## 871  56  28  13   7   4   8
```

Generalized regression with R - more functions

- Logistic model with Probit link:

```
probitmod <- glm(children ~ age,
    family=binomial(link=probit))
```

- Regression with count data:

```
modp <- glm(Tatoos ~ age,family=poisson)
```

- Proportional odds logistic regression in library `MASS`:

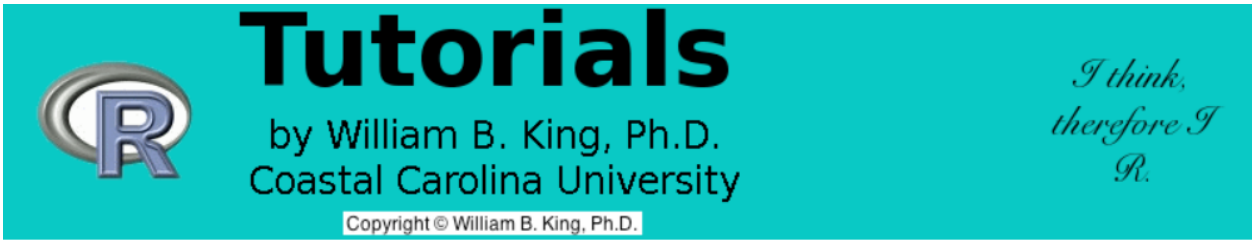


Figure 66:

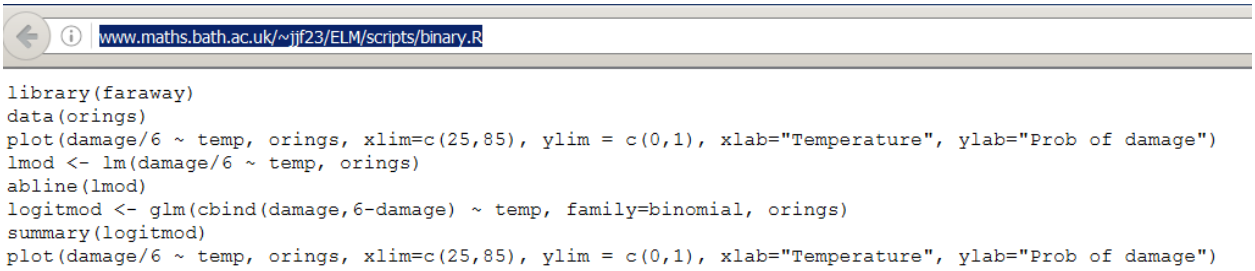


Figure 67:

```
library("MASS")
mod_plr<-polr(a11c020a ~ a11d096b ,data=dat)
```

B4A Exercise logistic regression

- Load the `MASS` package and combine `Pima.tr` and `Pima.tr2` to a data.frame called train and save Pima.te as test. Change the coding of our variable of interest to (type) to 0 (non-diabetic) and 1 (diabetic). Check for and take note of any missing values.

Linklist - logistic regression

- Introduction to logistic regression
- Code for the book of Faraway
- Categorical data:
- How to perform a Logistic Regression in R