RCourse: Day 2

Lukas Mödl, Matthias Becher, Erin Sprünken

Institut für Biometrie und Klinische Epidemiologie

Charité - Universitätsmedizin Berlin, Berlin

erin-dirk.spruenken@charite.de

November 9, 2022

Load

- `load()`
- `read.table()`
- `read.csv()`

Options of read.csv()

If we load CSV files in R, multiple parameters can be adjusted to tell R how the CSV is formatted. The most important parameters are:

- `header(TRUE/FALSE)`: Whether the first row contains variable names
- `sep`: What symbol is used as a separator. Default is ",". Often ";" or "\t"are used
- `dec`: What is the decimal separator, i.e. do we use "." or "," for decimal places
- Example: `read.csv("data.csv", header=TRUE, sep=";", dec=",")`

Indexing

Often we only want certain elements of a vector, list or data.frame. There are multiple solutions to that. The most straightforward one is to use indices directly. Consider the following vector:

```
x <- c(1, 2, 3, 4, 5)
```

- Choosing a certain value $\Leftrightarrow$ `x[1]`
- Choosing multiple values $\Leftrightarrow$ `x[c(1, 3, 5)]`
- Choose multiple contiguous values $\Leftrightarrow$ `x[1:3]`
- Leave a certain value out $\Leftrightarrow$ `x[-1]`

Indexing of Lists and Data Frames

List

- `x[1]`
- `x[[1]]`

Data Frame

- `x[1,]`
- `x[,1]`
- `x[,"Column1" ]`
- `x$Column1`

Summary()

```
    Numeric       Factor  Character            Logical              Date
Min.    :  1.00   a:25    Length:100         Mode :logical   Min.    :2022-01-01
1st Qu.: 25.75   b:25    Class :character   FALSE:50        1st Qu.:2022-01-25
Median : 50.50   c:25    Mode  :character   TRUE :50        Median :2022-02-19
Mean    : 50.50   d:25                                      Mean    :2022-02-19
3rd Qu.: 75.25                                             3rd Qu.:2022-03-16
Max.    :100.00                                            Max.    :2022-04-10
```

Functions for Descriptive Statistics

- Mean = `mean()`
- Median = `median()`
- Minimum = `min()`
- Maximum = `max()`
- Standard Deviation = `sd()`

Remark: sd() und var() divide by `n-1`

- Variance = `var()`
- Quantile = `quantile()`
- Correlation = `cor()`
- Covariance = `cov()`
- Contingency Table = `table()`

Handling of NAs

When computing different statistics, NAs can pose a problem to that.

- Example: `mean(c(1,2,3,4,5,NA))` returns `NA`

We solve this by providing `na.rm = TRUE` as an additional argument.

- Example: `mean(c(1,2,3,4,5,NA), na.rm = TRUE)` returns 3

The function `is.na()` detects whether there are NAs in a vector, data frame etc.

Conversion of Data

- Numeric $\Leftrightarrow$ `as.numeric()`
- Character $\Leftrightarrow$ `as.character()`
- Factor $\Leftrightarrow$ `as.factor()`
- Date $\Leftrightarrow$ `as.Date()`
- Logical $\Leftrightarrow$ `as.logical()`

Filtering of Data

Often, we want to filter our data regarding certain criteria, e.g. if we want to analyze only female subjects or only patients above a certain age. R provides different options to do so.

- `==`
  - `data[data$Sex == "F",]`
- `%in%`
  - `data[data$Color %in% c("blue", "red"),]`
- `subset()`
  - `subset(data, Age < 50)`

Logical Operators

If we want to check for multiple conditions at the same time, we can connect them with a logical operator instead of filtering one after another. R understands the following operators:

- and = &
- or (inclusive) = |
- Not = !

Furthermore, there are functions simplifying operations:

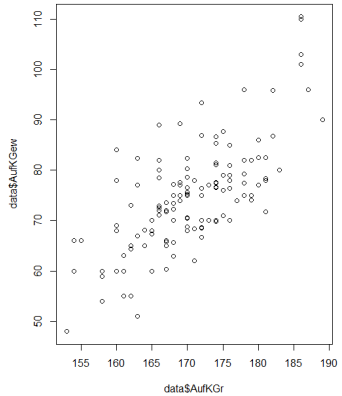
- all(x) $\Leftrightarrow$ checks, whether a logical vector contains only `TRUE`
- any(x) $\Leftrightarrow$ checks, whether a logical vector contains at least one `TRUE`
- which(x) *Leftrightarrow* checks, at which positions a logical vector contains x `TRUE`

Example:

- `data[data$Sex != "M" & data$Age > 50,]`

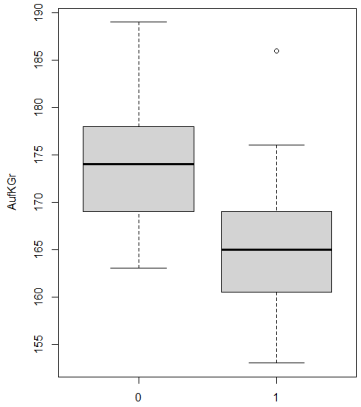
Scatterplot

- `plot(data$Height, data$Weight)`



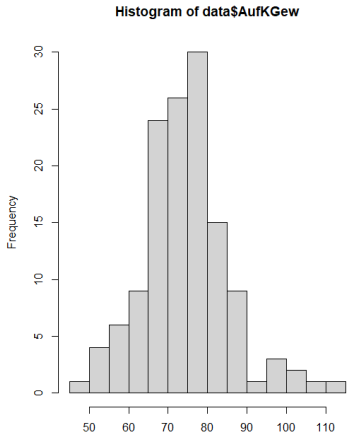
Boxplot

- `boxplot(ata$Sex == 0,]$Height, data[data$Sex == 1,]$Height)`



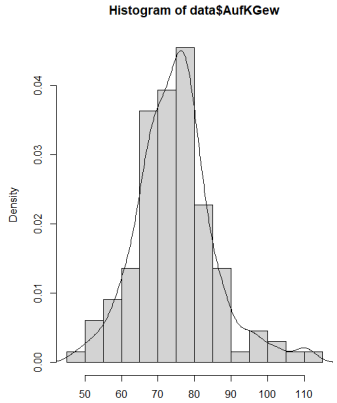
Histogram

- `hist(data$Weight)`



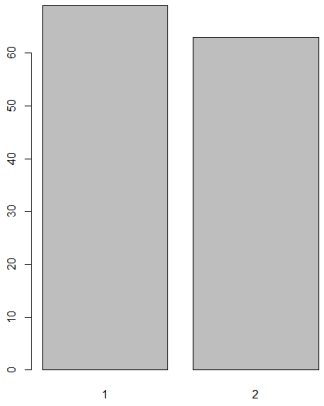
Histogram with Density

- `hist(data$Weight, probability = T)`
  `lines(density(data$Weight))`



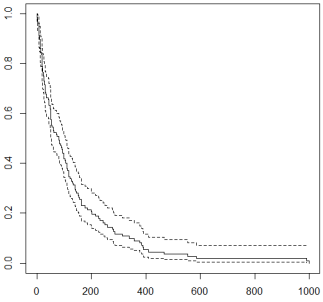
Barplot

- `barplot(summary(as.factor(data$Klinik)))`



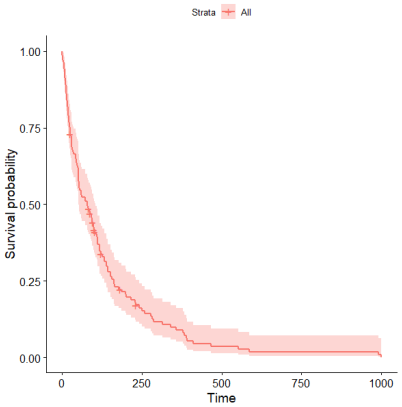
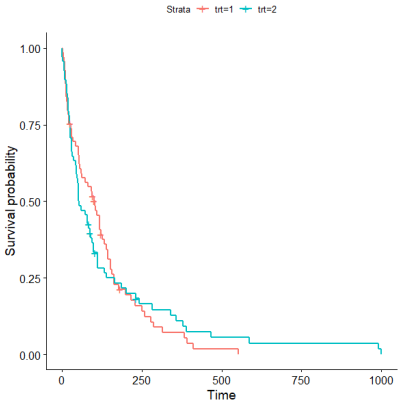
Kaplan-Meier Plot

```
library(survival)
data_vet <- veteran
km_fit <- survfit(Surv(time, status) ~ 1, data=data_vet)
plot(km_fit)
```



Kaplan-Meier Plot

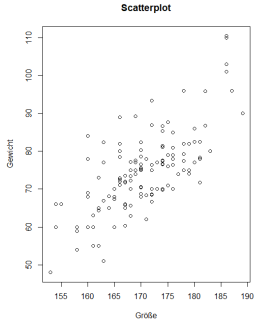
- ```
  library(survminer)
  ggsurvplot(km_fit)
  ```


  ```

Kaplan-Meier Plot

- ```
  km_fit <- survfit(Surv(time, status) ~ trt, data=data_vet)
  ggsurvplot(km_fit)
  ```


  ```

Labels for Plots

Plots are very flexibel and can be adjusted in many different ways. For example, we can include axis labels and a title.

- Beispiel:
```
plot(data$Height, data$Weight, xlab="Größe,"
     ylab="Gewicht", main="Scatterplot")
```



Saving of Plots