

Angewandte Datenanalyse mit R

Andreas Mock

Nationales Centrum für Tumorerkrankungen (NCT) Heidelberg

Sommersemester 2018



Hands-On Praktikum: Angewandte Datenanalyse mit R

Organisatorisches SS 2018

Kursunterlagen: <http://andreasmock.github.io/teaching>

Kontakt: andreas.mock@med.uni-heidelberg.de

Ort: K1, NCT 1.OG

Kurszeiten

- ▶ 05.06.2018 - 15:30 - 17:00 Uhr
- ▶ 12.06.2018 - **16:30 - 18:00 Uhr**
- ▶ 26.06.2018 - 15:30 - 17:00 Uhr

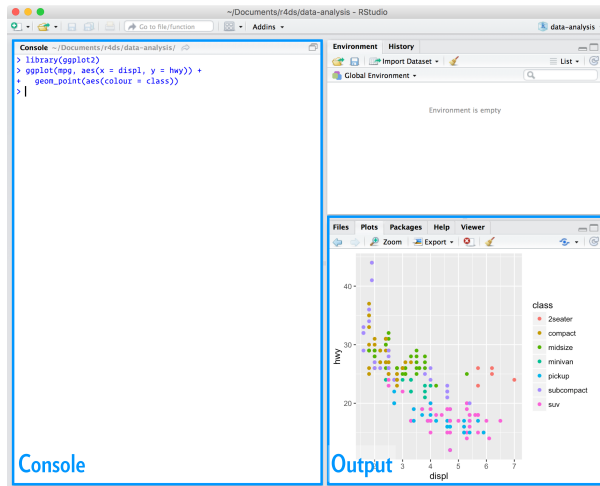
Translationale Onkologie 2018



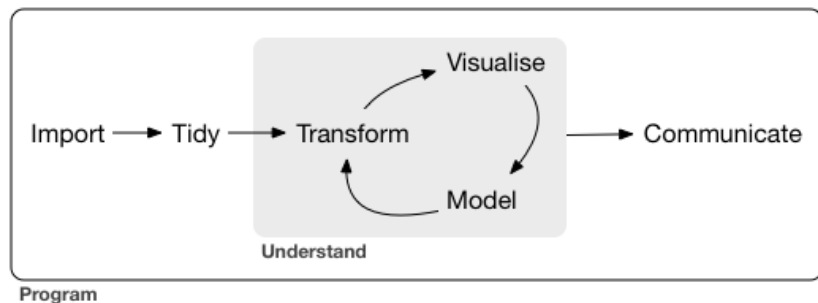
Was ist R?

- ▶ R ist eine **freie Programmiersprache** für statistische Berechnungen und Grafiken.
- ▶ Obwohl R bereits alt ist (Erscheinungsjahr 1993) gilt diese als **Standardsprache** für statistische Problemstellungen in Wirtschaft und Wissenschaft
- ▶ > **11.000 Formelsammlungen** für spezifische Fragestellungen (sog. Pakete)
- ▶ Hoch-qualitative und individuelle **Grafiken** - viele Wissenschaftler benutzen R nur hierzu
- ▶ Sowohl R, als auch alle Pakete sind **kostenlos!!**

Grafische Benutzeroberfläche und Entwicklungsumgebung für R



Data Science



R for Data Science, Hadley Wickham & Garrett Golemund 2016

Weiterführende Literatur

Programmieren mit R

Hands-On Programming with R, Garrett Golemund

- Kapitel 1, 2, 3, 4, 5, 7, 9

[Link](#)

Die “Bibel” für R User

R for Data Science, Hadley Wickham & Garrett Golemund

[Link](#)

Kochbuch für R Plots

R Graphics Cookbook, Winston Chang

[Link](#)

Installations- und Kaffeepause

R Coding 101

R Architektur

R ist eine Objekt-orientierte Programmiersprache. Es dreht sich daher im Grunde alles darum Objekte herzustellen, zu manipulieren und zu visualisieren.

```
# Zuweisung von Zahlen zu einem Objekt  
alter <- 67
```

R ist *case-sensitive* - Groß- und Kleinschreibung ist wichtig!

```
# Objekt `alter` ausgeben  
alter
```

```
## [1] 67
```

```
# .. entspricht nicht  
Alter
```

```
## Error in eval(expr, envir, enclos): object 'Alter' not found
```

R Objekttypen

Vektor

Def: Sammlung mehrerer Objekte gleicher Art (Länge 1 ist möglich).

```
# numerischer Vektor mit Länge 1  
x1 <- 5  
  
# Charaktervektor mit Länge 1 (wird in "" gesetzt)  
x2 <- "green"  
  
# Vektoren mit Länge 3  
y1 <- c(1,3,9)  
y2 <- c("gene1", "gene2", "gene3")
```

Subsetting:

```
# um den vollständigen Vektor y2 auszugeben  
y2  
  
## [1] "gene1" "gene2" "gene3"  
  
# um nur die ersten beiden Einträge des Vektors y2 auszugeben  
y2[1:2]  
  
## [1] "gene1" "gene2"
```

R Objekttypen

Matrix

Kombination mehrerer Vektoren gleichen Typs (numerisch oder Charakter). Die Matrix kann Zeilen- und Spaltennamen haben.

```
matrix <- cbind(y1, y1, y1)
rownames(matrix) <- y2
colnames(matrix) <- c("sample1", "sample2", "sample3")
matrix
```

```
##      sample1 sample2 sample3
## gene1      1      1      1
## gene2      3      3      3
## gene3      9      9      9
```

R Objekttypen

Matrix

Ein Subset kann man sich mit der folgenden Syntax anzeigen lassen:

```
matrix[Zeile,Spalte]
```

Bespiele hierfür sind:

```
matrix[1,]
```

```
## sample1 sample2 sample3  
##      1      1      1
```

```
matrix[,3]
```

```
## gene1 gene2 gene3  
##      1      3      9
```

```
matrix[1:2,]
```

```
##      sample1 sample2 sample3  
## gene1      1      1      1  
## gene2      3      3      3
```

R Objekttypen

Dataframe

Im Gegensatz zu Matrizen können in *Dataframes* Vektoren verschiedener Typen (z.B. numerischer Vektor und Charaktervektor) miteinander kombiniert werden. Wichtig: Die Vektoren müssen die gleiche Länge haben.

```
df <- data.frame(age=c(50,47,87),  
                  gender=c("male","male","female"))  
df
```

```
##   age gender  
## 1  50   male  
## 2  47   male  
## 3  87 female
```

Somit eignen sich *Dataframes* insbesondere für die Analyse von Patientenmetadaten im Rahmen von molekularbiologischen Experimenten oder klinischen Studien.

R Objekttypen

Vektor



1 Spalte bzw. Reihe
1 Typ (numerisch oder Text)

Matrix



Multiple Spalten / Reihen
1 Typ (numerisch oder Text)

Dataframe



Multiple Spalten / Reihen
mehrere Typen (z.B. numerisch
und Text)

R Objekttypen

Funktionen

Die Grundsyntax einer jeden Funktion ist

```
function(Objekt, Argumente)
```

Die Argumente sind hierbei fakultativ. R besitzt eine Vielzahl von Funktionen, ohne dass zusätzliche Pakete geladen werden müssen.

```
sum(y1)
```

```
## [1] 13
```

```
mean(y1)
```

```
## [1] 4.333333
```

Die Funktion `help` öffnet die Dokumentation in RStudio und zeigt die notwendigen Objekte und Argumente zu jeder Funktion an. Als Beispiel, was genau macht die Funktion `cbind`?

```
help(cbind)
```


Bespieldaten des Kurses

Metadaten des *The Cancer Genome Atlas (TCGA)* zur Analyse von Kopf-Hals-Tumoren (head and neck squamous cell carcinoma; HNSCC). Der Datensatz fasst die wichtigsten klinisch-pathologischen Charakteristika der Studienkohorte (n=279) zusammen.

[Link zur Originalpublikation](#)

Bespieldaten des Kurses

```
head(hnsc)
```

```
##           id age alcohol days_to_death gender  neoplasm_site grade
## 1 TCGA-BA-4074 69     YES           461   MALE    Oral Tongue   G3
## 2 TCGA-BA-4076 39     YES           415   MALE      Larynx     G2
## 3 TCGA-BA-4077 45     YES          1134 FEMALE Base of Tongue  G2
## 4 TCGA-BA-4078 83     NO            276   MALE      Larynx     G2
## 5 TCGA-BA-5149 47     YES           248   MALE Floor of Mouth   G2
## 6 TCGA-BA-5151 72     YES           190   MALE  Buccal Mucosa   G1
##   pack_years                tobacco_group tumor_stage
## 1          51                Current smoker   Stage IVA
## 2          30                Current smoker         <NA>
## 3          30 Current reformed smoker for < or = 15 years   Stage IVA
## 4          75 Current reformed smoker for < or = 15 years         <NA>
## 5          60                Current smoker   Stage IVA
## 6          20      Current reformed smoker for > 15 years   Stage IVA
##   vital_status
## 1    DECEASED
## 2    DECEASED
## 3    DECEASED
## 4    DECEASED
## 5     LIVING
## 6     LIVING
```

Fragen?