Wichtige Befehle in R zur Datenanalyse

Ergänzende Unterlagen zur Vorlesung

Prof. Dr. Oliver Gansser

Institut für Empirie & Statistik
der FOM Hochschule
für Oekonomie & Management

Inhaltsverzeichnis

Einleitung

Das Programm R

Warum R?

- R ist ein Programm für Statistik und Datenanalyse.
- R ist für Linux, MacOS X und Windows (95 oder höher) Plattformen verfügbar.
- R ist eine elegante und umfassende statistische und grafische Programmiersprache.
- R kann eine steile Lernkurve L haben `(L = Zeiteinheit/Erfolgseinheit)`.
- R ist kostenlos! Wenn Sie Lehrender oder Studierender sind, sind die Vorteile offensichtlich.
- R bietet eine unvergleichliche Plattform für die Programmierung neuer statistischer Methoden in einer einfachen und unkomplizierten Weise.
- R enthält fortgeschrittene statistische Routinen, die noch nicht in anderen Software-Paketen verfügbar sind.
- R verfügt über state-of-the-art Grafiken Fähigkeiten.

Warum RStudio?

RStudio ist eine integrierte Entwicklungsumgebung (IDE), die die Verwendung von R für Anfänger und Experten erleichtert.

Warum RCommander?

Der RCommander ist eine grafische Benutzeroberfläche, allgemein bekannt als Rcmdr, und kann als Paket in R und RStudio geladen werden.

Wo bekomme ich R und R Studio?

- Die Software R kann von einer der CRAN-Spiegelservern unter https://cran.r-project.org/mirrors.html heruntergeladen werden.
- RStudio bekommen Sie auf der Homepage von RStudio unter https://www.rstudio.com.

Arbeiten mit R und RStudio oder RCommander

- Mit RStudio benutzen Sie entweder die Console oder ein Skiptfenster zur Eingabe der Skriptbefehle. Ein neues R-Skriptfenster öffen Sie über **File–>New File–>R Script**.
- Falls Sie den RCommander benutzen, können Sie sämtliche Skriptbefehle in das Skriptfenster (oberes Fenster) eingeben.
- Wenn Sie ein RStudio-Skipt benutzen, dann gibt es vier Fenster:
  - links oben die das Skript,
  - links unten die Console,
  - rechts oben Daten, Objekte und Historie und
  - links unten Dateien, Abbildung, Hilfeseiten und Tipps

- Einzelne Befehle aus dem Skriptfenster in R Studio können Sie anstatt mit dem **Run Button** auch mit **Str** und **Enter** an die Console schicken.
- Befehlen können auch direkt in der Console ausgeführt werden (Ausführung mi der Enter-Taste).
- Das Arbeiten mit Skript hat einige Vorteile und macht die Datenanalyse sehr komfortabel.

Funktionalitäten in R

- Zusätzliche Funktionalitäten können über Zusatzpakete hinzugeladen werden. Diese müssen ggf. zunächst installiert werden.
- Paketen könne über den Menüpunkt **Tools** installiert werden oder durch den Befehl **install.packages("Names des Pakets")**.
- Pakete werden geladen mit dem Befehl **library(Name des Pakets)** oder **require(Name des Pakets)**.
- Achten Sie auch die **Anführungszeichen** nur beim Installieren.
- Mit der Pfeiltaste nach oben können Sie (nur in der Console) einen vorherigen Befehl wieder aufrufen.
- Im R-Skrip können Sie sich die Befehle und Erläuterungen individuell einrichten und diese unabhängig von der Reihenfolge immer wieder ausführen.

Umgang mit Daten in R

Das Bearbeiten und Ändern von Daten funktioniert mit R deutlich anders als mit SPSS - schon weil es keine graphische Oberfläche zum Bearbeiten der Daten gibt. Das ist aber kein Nachteil, sondern ein Vorteil.

Der typische SPSS Datensatz mit Variablen in den Spalten und Fällen (Versuchspersonen) in den Zeilen heißt bei R **data frame** und ist ein Objekt im Workspace. Ein Objekt kann entweder durch das Einlesen einer Datendatei oder durch Zuweisung von Daten erzeugt werden. Zum Einsatz kommt dabei der Zuweisungsbefehl **<-**.

Datenmanagement in R

Woher die Daten nehmen?

Daten können auf unterschiedliche Art und Weise generiert werden:

- Erzeugen eigenen Daten
- Einlesen von Sekundärdaten.
- Einlesen von Daten aus Umfragestudie mittels Befragungssoftware oder händischer Eingabe.

Eigene Daten erzeugen

```
# Zwei Vektor erzeugen
Bewertung<-c(8,10,5,6,2) # nummerische Daten
Geschlecht<-c("m","w","m",NA,"w") #kategoriale Daten
```

```
# data frame erzeugen
data<-data.frame(Bewertung,Geschlecht)

# Gesamten data frame ausgeben
data
```

```
##   Bewertung Geschlecht
## 1         8          m
## 2        10          w
## 3         5          m
## 4         6       <NA>
## 5         2          w
```

```
# Nur eine Spalte ausgeben
data[,2] # Spaltennummer reicht aus
```

```
## [1] m    w    m    <NA> w
## Levels: m w
```

```
# Eine bestimmte Zeile ausgeben
data[3,]
```

```
##   Bewertung Geschlecht
## 3         5          m
```

```
# Inhalte einer Variablen fortlaufend ausgeben
data$Geschlecht
```

```
## [1] m    w    m    <NA> w
## Levels: m w
```

```
# Variablen ändern (z. B. recodieren der 10er Skala)
data$Bewertung <- 11-data$Bewertung # (n+1)-X recodiert eine beliebige Skala
data[1]
```

```
##   Bewertung
## 1         3
## 2         1
## 3         6
## 4         5
## 5         9
```

```
data$Bewertung
```

```
## [1] 3 1 6 5 9
```

Nützliche Funktionen in R

```
length(data) # Anzahl der Elemente oder Komponenten
```

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

```
str(data) # Struktur der Daten
```

```
## 'data.frame':    5 obs. of  2 variables:
##  $ Bewertung : num  3 1 6 5 9
##  $ Geschlecht: Factor w/ 2 levels "m","w": 1 2 1 NA 2
```

```
class(data) # Klasse oder Typ der Daten
```

```
## [1] "data.frame"
```

```
names(data) # Variablennamen
```

```
## [1] "Bewertung"  "Geschlecht"
```

```
c(1:30) # kombinieren Daten in einen Vektor
```

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

```
cbind(Bewertung, Geschlecht) # kombiniere die Daten als Spalten
```

```
##      Bewertung Geschlecht
## [1,] "8"       "m"
## [2,] "10"      "w"
## [3,] "5"       "m"
## [4,] "6"       NA
## [5,] "2"       "w"
```

```
rbind(Bewertung, Geschlecht) # kombiniere Daten als Zeilen
```

```
##            [,1] [,2] [,3] [,4] [,5]
## Bewertung  "8"  "10" "5"  "6"  "2"
## Geschlecht "m"  "w"  "m"  NA   "w"
```

```
ls()      # Gibt alle aktiven Objekte aus
```

```
## [1] "Bewertung"  "data"       "Geschlecht"
```

```
rm(data) # Löscht Objekte
data<-data.frame(Bewertung,Geschlecht) # und erzeugt sie wieder
```

```
newdata <- edit(data)    # Editiert ein Objekt und speichert unter neuem Namen
fix(data)                # edit in place
```

Arbeiten mit einem neuen Datensatz

Daten einlesen

Es ist praktisch, wenn die eingelesenen Daten im gleichen Verzeichnis liegen wie das Skript. Normalerweise arbeitet man nicht nur einmal mit einem Datensatz, sondern immer wieder und lange Zeit und auf unterschiedlichen Computern. Das Skript beinhaltet dabei alle Informationen: * Welcher Datensatz wird eingelesen. * Was wird an den Daten verändert. * Welche Verfahren werden mit dem eingelesenen Datensatz durchgeführt. * et cetera pp.

Wir verwenden für die Übungen den `tips Datensatz`. Dazu laden wir die Daten auf unseren Rechner und importieren die Daten.

```
# Download der Daten mit dem Befehl
download.file("https://goo.gl/whKjnl", destfile = "tips.csv")
tips<-read.csv2("tips.csv")
```

Wenn die Daten nicht im gleichen Pfad wie das Arbeitsverzeichnis liegen, dann muss das Arbeitsverzeichnis gesucht werden. Workspace suchen mit getwd().

```
getwd()
```

Wenn die Daten aus einem anderen Pfad als dem Workspace geladen werden sollen, muss der Pfad angegeben werden:

setwd(“C:/.../...”)

Datenstruktur prüfen

Es empfiehlt sich, vor jeder Analyse den Datensatz zu betrachten. Von besonderer Relevanz ist hier die Datenstruktur, d.h. Variablennamen und Variablentypen, die Größe des Datensatzes sowie die ersten und letzten paar Zeilen.

```
# Datenstruktur betrachten
str(tips)
```

```
## 'data.frame':    244 obs. of  7 variables:
##  $ total_bill: num  17 10.3 21 23.7 24.6 ...
##  $ tip       : num  1.01 1.66 3.5 3.31 3.61 4.71 2 3.12 1.96 3.23 ...
##  $ sex       : Factor w/ 2 levels "Female","Male": 1 2 2 2 1 2 2 2 2 2 ...
##  $ smoker    : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ day       : Factor w/ 4 levels "Fri","Sat","Sun",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ time      : Factor w/ 2 levels "Dinner","Lunch": 1 1 1 1 1 1 1 1 1 1 ...
##  $ size      : int  2 3 3 2 4 4 2 4 2 2 ...
```

```
# Dimensionen des Datensatzes
dim(tips)
```

```
## [1] 244   7
```

```
# Kopf oder Ende (tail) der Datenmatrix betrachten
head(tips)
```

```
##   total_bill  tip    sex smoker day   time size
## 1      16.99 1.01 Female     No Sun Dinner    2
## 2      10.34 1.66   Male     No Sun Dinner    3
## 3      21.01 3.50   Male     No Sun Dinner    3
## 4      23.68 3.31   Male     No Sun Dinner    2
## 5      24.59 3.61 Female     No Sun Dinner    4
## 6      25.29 4.71   Male     No Sun Dinner    4
```

```
tail(tips)
```

```
##     total_bill  tip    sex smoker  day   time size
## 239      35.83 4.67 Female     No  Sat Dinner    3
## 240      29.03 5.92   Male     No  Sat Dinner    3
## 241      27.18 2.00 Female    Yes  Sat Dinner    2
## 242      22.67 2.00   Male    Yes  Sat Dinner    2
## 243      17.82 1.75   Male     No  Sat Dinner    2
## 244      18.78 3.00 Female     No Thur Dinner    2
```

```
# Ausgabe der levels bei kategorialen Variablen
levels(tips$sex)
```

```
## [1] "Female" "Male"
```

Datenstruktur verändern

Normalerweise liegen neu erfasste Daten nicht so vor, wie wir sie für die Datenanalyse benötigen. Wir müssen eventuell Daten konvertieren und löschen. Beispielweise wird Geschlecht über ein Online-Befragungstool meist nur nummerisch erfasst und die Zuweisung muss dann im Rahmen der Datenbereinigung und -aufbereitung erfolgen. Wie wir Variablen recodieren, haben wir bereits weiter oben gelernt.

```
# Das Geschlecht im Datensatz tips liegt bereits kategorial vor
tips$sex
```

```
##   [1] Female Male   Male   Male   Female Male   Male   Male   Male   Male
##  [11] Male   Female Male   Male   Female Male   Female Male   Female Male
##  [21] Male   Female Female Male   Male   Male   Male   Male   Male   Female
##  [31] Male   Male   Female Female Male   Male   Male   Female Male   Male
```

```
##  [41] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
##  [51] Male   Female Female Male   Male   Male   Male   Female Male   Male
##  [61] Male   Male   Male   Male   Male   Male   Female Female Male   Male
##  [71] Male   Female Female Female Female Male   Male   Male   Male   Male
##  [81] Male   Male   Female Male   Male   Female Male   Male   Male   Male
##  [91] Male   Male   Female Female Female Male   Male   Male   Male   Male
## [101] Female Female Female Female Female Male   Male   Male   Male   Female
## [111] Male   Female Male   Male   Female Female Male   Female Female Female
## [121] Male   Female Male   Male   Female Female Male   Female Female Male
## [131] Male   Female Female Female Female Female Female Female Male   Female
## [141] Female Male   Male   Female Female Female Female Female Male   Male
## [151] Male   Male   Male   Male   Male   Female Male   Female Female Male
## [161] Male   Male   Female Male   Female Male   Male   Male   Female Female
## [171] Male   Male   Male   Male   Male   Male   Male   Male   Female Male
## [181] Male   Male   Male   Male   Male   Male   Female Male   Female Male
## [191] Male   Female Male   Male   Male   Male   Male   Female Female Male
## [201] Male   Female Female Female Male   Female Male   Male   Male   Female
## [211] Male   Male   Male   Female Female Female Male   Male   Male   Female
## [221] Male   Female Male   Female Male   Female Female Male   Male   Female
## [231] Male   Male   Male   Male   Male   Male   Male   Male   Female Male
## [241] Female Male   Male   Female
## Levels: Female Male
```

```
# Ausgabe der levels
levels(tips$sex)
```

```
## [1] "Female" "Male"
```

```
# Konvertieren von factor auf nummerisch
tips$sex<-as.numeric(tips$sex)
tips$sex
```

```
##   [1] 1 2 2 2 1 2 2 2 2 2 2 1 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 2 1 2 2 1 1 2
##  [36] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 1 2 2
##  [71] 2 1 1 1 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 1 1 1 2 2 2 2 2 1 1 1 1 1
## [106] 2 2 2 2 1 2 1 2 2 1 1 2 1 1 1 2 1 2 2 1 1 2 1 1 2 2 1 1 1 1 1 1 1 2 1
## [141] 1 2 2 1 1 1 1 1 2 2 2 2 2 2 2 2 1 2 1 1 2 2 2 1 2 1 2 2 2 2 1 1 2 2 2 2 2
## [176] 2 2 2 1 2 2 2 2 2 2 2 2 1 2 1 2 2 1 2 2 2 2 2 1 1 2 2 1 1 1 2 1 2 2 2 1
## [211] 2 2 2 1 1 1 2 2 2 1 2 1 2 1 2 1 1 2 2 1 2 2 2 2 2 2 2 2 1 2 1 2 2 1
```

```
levels(tips$sex) # Es gibt keine, da jetzt nummerisch!
```

```
## NULL
```

```
# Konvertieren von nummerisch auf factor
tips$sex<-as.factor(tips$sex)
levels(tips$sex)
```

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

```
# Wertzuweisung von levels
levels(tips$sex)<-c("f","m")
levels(tips$sex)
```

```
## [1] "f" "m"
```

```
tips$sex
```

```
##   [1] f m m m f m m m m m m f m m f m f m f m m f f m m m m m f m m f f m
##  [36] m m f m m m m m m m m m m m m m m f f m m m m f m m m m m m m f f m m
##  [71] m f f f f m m m m m m m f m m f m m m m m m f f f m m m m m f f f f f
## [106] m m m m f m f m m f f m f f f m f m m f f m f f m m f f f f f f f m f
## [141] f m m f f f f f m m m m m m m f m f f m m m f m f m m m f f m m m m m
## [176] m m m f m m m m m m m m f m f m m f m f m m m m m f f m m f f f m f m m m f
## [211] m m m f f f m m m f m f m f m f f m m f m m m m m m m m f m f m m f
## Levels: f m
```

```
#oder alternativ alles in einem Schritt
tips$sex<-factor(tips$sex, labels = c("female", "male")) # sVariable sex wir wieder übe
levels(tips$sex)
```

```
## [1] "female" "male"
```

Arbeiten mit dem Paket **mosaic**

Die folgende Syntax

Zielbefehl(y ~ x | z, data=...)

wird verwendet für

- graphische Zusammenfassungen,
- numerische Zusammenfassungen und
- inferentstatistische Auswertungen

Für Grafiken gilt:

- y: y-Achse Variable
- x: x-Achse Variable
- z: Bedingungsvariable

Generell gilt:

```
y ~ x | z
```

kann in der Regel gelesen werden **y wird modelliert von (oder hängt ab von) x unterschieden für jedes z**.

Laden der mosaic Paktes

```
require(mosaic)
```

Literaturempfehlung für den Einstieg in R mit dem Paket mosaic

- Daniel T. Kaplan, Nicholas J. Horton, Randall Pruim, (2013): Project MOSAIC Little Books *Start Teaching with R*, http://mosaic-web.org/go/Master-Starting.pdf

Versionshinweise:

- Datum erstellt: 2017-08-03
- R Version: 3.4.0
- `mosaic` Version: 0.14.4