R Kurzreferenz

Karsten Lübke

Vorbemerkungen

Eine Übersicht von nützlichen R Funktionen innerhalb der Datenanalyse.

Diese Kurzreferenz beschreibt ein kleinen Teil der R Funktionen, wobei größtenteils auf das Zusatzpaket `mosaic` zurückgegriffen wird. Sie basiert größtenteils auf der Vignette Minimal R von Randall Pruim.

Weitere Hilfe und Beispiele finden Sie, wenn Sie

```
?Befehl
```

eingeben.

- R unterscheidet zwischen Groß- und Kleinbuchstaben
- R verwendet den Punkt `.` als Dezimaltrennzeichen
- Fehlende Werte werden in R durch `NA` kodiert
- Kommentare werden mit dem Rautezeichen `#` eingeleitet; der Rest der Zeile von von R dann ignoriert.
- R wendet Befehle direkt an
- R ist objektorientiert, d. h. dieselbe Funktion hat evtl. je nach Funktionsargument unterschiedliche Rückgabewerte
- Zusätzliche Funktionalität kann über Zusatzpakete hinzugeladen werden. Diese müssen ggf. zunächst installiert werden
- Mit der Pfeiltaste nach oben können Sie einen vorherigen Befehl wieder aufrufen
- Eine Ergebniszuweiseung erfolgt über `<-`

Innerhalb von `mosaic`:

```
analysiere(y ~ x | z , data=Daten)
```

d. h., modelliere `y` in Abhängigkeit von `x` getrennt für jedes `z` aus dem Datensatz `Daten`. Dabei können Teile fehlen.

Zusatzpakete müssen vor der ersten Benutzung einmalig installiert und geladen werden:

```
install.packages("Paket") # Einmalig installieren
require(Paket) # Laden, einmalig in jeder Sitzung
```

Daten

Daten einlesen und Datenvorverarbeitung sind häufig der (zeitlich) aufwendigste Teil einer Datenanalyse. Da die Daten die Grundlage sind, sollte auch hier sorgfältig gearbeitet und überprüft werden.

Daten einlesen

```
read.table() # Allgemeinen Datensatz einlesen. Achtung: Optionen anpassen
read.csv2() # (deutschen) csv Datensatz einlesen
file.choose() # Datei auswählen
meineDaten <- read.csv2(file.choose())
```

U. a. mit Hilfe des Zusatzpaketes `readxl` können Excel Dateien eingelesen werden:

```
meineDaten <- read_excel(file.choose())
```

Daten verarbeiten

```
str() # Datenstruktur
head() # Obere Zeilen
tail() # Untere Zeilen
nrow(); ncol() # Anzahl Zeilen; Spalten
rownames(); colnames() # Zeilennamen, Spaltennamen
```

Daten transformieren

Einzelne Variablen eines Datensatzes können über `$` ausgewählt werden: `Daten$Variable`. Allgemein kann über `Daten[i,j]` die i. Zeile und j. Spalte ausgewählt werden, wobei auch mehrere oder keine Zeile(n) bzw. Spalten ausgewählt werden können. Über `c()` wird ein Vektor erzeugt.

```
as.factor() # Daten als Faktoren definieren
relevel() # Faktorstufen umordnen
as.numeric() # Faktorstufen als numerische Daten verwenden
subset() # Teilmenge der Daten auswählen
na.omit() # Zeilen mit fehlenden Werten entfernen
log() # Logarithmusfunktion
exp() # Exponentialfunktion
sqrt() # Quadratwurzelfunktion
abs() # Betragsfunktion
cut() # Aufteilung numerischer Werte in Intervalle
rowSums() # Zeilensumme
rowMeans() # Zeilenmittelwert
recode() # Umkodierung von Werten, paket car
```

Grafische Verfahren

Vor jeder mathematisch-statistischen Analyse sollte eine explorative, grafische Analyse erfolgen. Die folgenden Befehle sind aus dem Paket `mosaic`.

```
bargraph() # Balkendiagramm
histogram() # Histogramm
bwplot() # Boxplot
xyplot() # Streudiagrmm
```

Nicht aus dem Paket mosaic sind:

```
mosaicplot() # Mosaicplot
corrgram() # Korrelationsplot, Paket corrgram
ggpairs() # Matrixplot, GGally
heatmap() # Heatmap
```

Deskriptive Statistik

Eine gute Zusammenfassung liefert der `mosaic` Befehl:

```
favstats()
```

Ansonsten (`mosaic` angepasst):

```
tally() # Tabellierung, Häufigkeiten
mean() # Arithmetischer Mittelwert
median() # Median
sd() # Standardabweichung
```

```
var() # Varianz
IQR() # Interquartilsabstand
cov() # Kovarianz
cor() # Korrelationskoefizient
```

Induktive Statistik

Randomisierung, Simulationen

Größtenteils `mosaic`:

```
set.seed() # Zufallszahlengenerator setzen
rflip() # Münzwurf
do() # Wiederholung (Schleife)
sample() # Stichprobe ohne Zurücklegen
resample() # Stichprobe mit Zurücklegen
rnorm() # Normalverteilte Zufallszahlen
```

Verteilungen

Innerhalb der Funktionen müssen ggf. die Parameter, d. h. `mean=`, `sd=`, bzw. `df=` angepasst werden.

```
pchisq() # Verteilungsfunktion Chi² Verteilung
qchisq() # Quantilsfunktion Chi² Verteilung
pnorm() # Verteilungsfunktion Normalverteilung
qnorm() # Quantilsfunktion Normalverteilung
pt() # Verteilungsfunktion t-Vverteilung
qt() # Quantilsfunktion t-Vverteilung
```

Analoger Aufbau für weitere Verteilungen, z. B. `_pniom()`, `_f()`.

Testverfahren

Einige der Testverfahren wurden von `mosaic` angepasst.

```
t.test() # t-Test
binom.test() # Binomialtest
xchisq.test() # Chi²-Test
aov() # Varianzanalyse
```

Der Nicht-parametrische Wilcoxon-Test `wilcox.test()` ist nicht im Paket `mosaic`, hat daher einen leicht anderen Funktionsaufruf. Einen Test auf Normalverteilung führt der Shapiro-Wilk Test durch: `shapiro.test()`.

Multivariate Verfahren

```
lm() # Lineare Regression
glm(, family="binomial") # Logistische Regression
residuals() # Residuen einer Regression
fitted() # Angepasste Werte einer Regression
```

In `mosaic` kann das Ergebnis einer solchen Regression über `makeFun()` in eine einfache mathematische Funktion überführt werden. `plotFun()` zeichnet das Ergebnis. `step()` führt eine Variablenselektion durch.

Weitere Verfahren - nicht `mosaic`:

```
kmeans() # k-Means Clusterverfahren
prcomp() # Hauptkomponentenanalyse (PCA)
rpart() # Klassifikations- und Regressionsbäume, Paket rpart
```

Versionshinweise:

- Datum erstellt: 2016-06-27
- R Version: 3.3.1
- `mosaic` Version: 0.14