R Kurzreferenz

Autor*innen:

Prof. Dr. Karsten Lübke

Stand SoSe 2019

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 Befehle haben Ihre Argumente immer innerhalb von Klammern, und Optionen etc. werden durch Komma getrennt
- 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 in der Konsole wieder aufrufen
- Eine Ergebniszuweisung erfolgt über `<-`
- Ein `+` heißt, das R auf eine weitere Eingabe wartet (Anführungsstriche, Klammer zu ...)
- Befehle können über `Esc` abgebrochen werden

Innerhalb von `mosaic`:

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

d. h., modelliere `y` in Abhängigkeit von `x` getrennt bzw. bedingt für `z` aus dem Datensatz `Daten`. Dabei können Teile (z. B. `y` und/ oder `z`) fehlen.[1]

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

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

[1] Beim Mac ist `~` die Tastenkombination `alt+n`, `|` die Tastenkombination `alt+7`

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() # csv Datensatz einlesen (aus deutschsprachigem Excel)
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
inspect() # Datenübersicht
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. Spalte(n) ausgewählt werden können. Über `c()` wird ein Vektor erzeugt.

```
as.factor() # Daten als Faktoren definieren
relevel() # Faktorstufen umordnen
droplevels() # Ungenutzte Faktorstufen entfernen
recode() # Umkodierung von Werten, Paket car
as.numeric() # Faktorstufen als numerische Daten verwenden
cut() # Aufteilung numerischer Werte in Intervalle

subset() # Teilmenge der Daten auswählen
na.omit() # Zeilen mit fehlenden Werten entfernen

log() # Logarithmusfunktion
```

```r
exp() # Exponentialfunktion
sqrt() # Quadratwurzelfunktion
abs() # Betragsfunktion

rowSums() # Zeilensumme
rowMeans() # Zeilenmittelwert
```

Innerhalb des Paketes `dplyr` (wird mit `mosaic` geladen) gibt es u. a. folgende Funktionen:

```r
filter() # Filtert Beobachtungen eines Datensatzes
select() # Wählt Variablen eines Datensatzes aus
mutate() # Erzeugt neue Variable bzw. verändert bestehende
arrange() # Sortiert Beobachtungen eines Datensatzes
mutate() # Variablen verändern, neu erzeunge
summarise() # Beobachtungen zusammenfassen
group_by() # Beobachtungen gruppieren
%>% # Übergebe das Ergebnis der vorhergehenden Funktion an die folgende
```

Grafische Verfahren

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

```r
gf_bar()  # Balkendiagramm (Anzahl)
gf_percents  # Balkendiagramm (Prozentangabe)
gf_props  # Balkendiagramm (Anteilswertsangabe)
gf_histogram()  # Histogramm
gf_boxplot()  # Boxplot
gf_point()  # Streudiagrmm
```

Nicht aus dem Paket `mosaic` sind:

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

Deskriptive Statistik

Eine gute Zusammenfassung numerischer Variablen liefert der `mosaic` Befehl:

```
favstats()
```

Ansonsten (`mosaic` angepasst):

```
tally() # Tabellierung, Häufigkeiten
prop() # Anteile
diffprop() # Differenz Anteile
mean() # Arithmetischer Mittelwert
diffmean() # Differenz Mittelwerte
median() # Median
quantile() # Quantile
sd() # Standardabweichung
var() # Varianz
IQR() # Interquartilsabstand
cov() # Kovarianz
cor() # Korrelationskoefizient
```

Inferenzstatistik

Randomisierung, Simulationen

Größtenteils `mosaic`:

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

Verteilungen

Innerhalb der Funktionen müssen ggf. die Parameter, d.h. `mean=`, `sd=`, bzw. `df=` angepasst werden. (Das vorgestellte `x` steht für in `mosaic` angepasste Versionen.)

```
xpchisq() # Verteilungsfunktion Chi² Verteilung
xqchisq() # Quantilsfunktion Chi² Verteilung
xpnorm() # Verteilungsfunktion Normalverteilung
xqnorm() # Quantilsfunktion Normalverteilung
xpt() # Verteilungsfunktion t-Verteilung
xqt() # Quantilsfunktion t-Verteilung
```

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

Testverfahren

Einige der Testverfahren wurden von `mosaic` angepasst.

```
t.test() # t-Test
prop.test() # Binomialtest (approximativ)
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()`. `cohensD()` aus dem Paket `lsr` berechnet die Effektgröße *Cohens D*.

Multivariate Verfahren

```
lm() # Lineare Regression
glm(, family="binomial") # Logistische Regression
plotModel() # Modell zeichnen
coef() # Koeffizienten extrahieren
residuals() # Residuen einer Regression
fitted() # Angepasste Werte einer Regression
predict() # Vorhersagen
summary() # Zusammenfassung
anova() # ANOVA Tabelle
```

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`:

```
prcomp() # Hauptkomponentenanalyse (PCA)
alpha() # Reliabilitätsanalys, Paket psych
dist() # Distanzen
hclust() # Hierachsiche Clusteranalyse
kmeans() # k-Means Clusterverfahren
rpart() # Klassifikations- und Regressionsbäume, Paket rpart
```

Weitere Themen

Disparitäts- und Konzentrationsmessung

Paket `ineq`:

```
Lc() # Loremzkurve
ineq() # Gini Koeffizient (u. a.)
```

Zeitreihen

```
ma() # Gleitende Durchschnitte, Paket forecast
stl() # Zeitreihe zerlegen
```

Versionshinweise:

- Datum erstellt: 2019-01-24
- R Version: 3.5.1
- `mosaic` Version: 1.4.0