Einführung in die Datenanalyse mit R - Datenanalyse

Jan-Philipp Kolb

3 Mai 2017

Streuungsmaße

Im base Paket sind die wichtigsten Streuungsmaße enthalten:

- Varianz: `var()`
- Standardabweichung: `sd()`
- Minimum und Maximum: `min()` und `max()`
- Range: `range()`

```
ab <- rnorm(100); var(ab)
```

```
## [1] 0.9836797
```

```
sd(ab); range(ab)
```

```
## [1] 0.9918063
```

```
## [1] -2.583554  2.413130
```

Extremwerte

```
min(ab)
```

```
## [1] -2.583554
```

```
max(ab)
```

```
## [1] 2.41313
```

Fehlende Werte

- Sind NAs vorhanden muss dies der Funktion mitgeteilt werden

```
ab[10] <- NA

var(ab)
```

```
## [1] NA
```

Bei fehlenden Werten muss ein weiteres Argument mitgegeben werden:

```
var(ab,na.rm=T)
```

```
## [1] 0.9872602
```

Häufigkeiten und gruppierte Kennwerte

- Eine Auszählung der Häufigkeiten der Merkmale einer Variable liefert `table()`
- Mit `table()` sind auch Kreuztabellierungen möglich indem zwei Variablen durch Komma getrennt werden: `table(x,y)` liefert Häufigkeiten von `y` für gegebene Ausprägungen von `x`

```
x <- sample(1:10,100,replace=T)

table(x)
```

```
## x
##  1  2  3  4  5  6  7  8  9 10
##  9 16  8  9 15  7  6 13  9  8
```

Tabellieren - weiteres Beispiel

```
musician <- sample(c("yes","no"),100,replace=T)
```

```
?table
```

```
table(x)
```

```
## x
##  1  2  3  4  5  6  7  8  9 10
##  9 16  8  9 15  7  6 13  9  8
```

```
table(x,musician)
```

```
##     musician
## x     no yes
##   1    7   2
##   2    9   7
```

Eine weitere Tabelle

```
data(esoph)
table(esoph$agegp)
```

```
##
## 25-34 35-44 45-54 55-64 65-74   75+
##    15    15    16    16    15    11
```

Häufigkeitstabellen

- `prop.table()` liefert die relativen Häufigkeiten
- Wird die Funktion außerhalb einer `table()` Funktion geschrieben erhält man die relativen Häufigkeiten bezogen auf alle Zellen

Die Funktion prop.table()

```
table(esoph$agegp,esoph$alcgp)
```

```
##
##           0-39g/day 40-79 80-119 120+
##    25-34          4     4       3    4
##    35-44          4     4       4    3
##    45-54          4     4       4    4
##    55-64          4     4       4    4
##    65-74          4     3       4    4
##    75+            3     4       2    2
```

Die aggregate Funktion

- Mit der `aggregate()` Funktion können Kennwerte für Untergruppen erstellt werden
- `aggregate(x,by,FUN)` müssen mindestens drei Argumente übergeben werden:

```
aggregate(state.x77,by=list(state.region),mean)
```

```
##         Group.1 Population   Income Illiteracy Life Exp    
## 1     Northeast   5495.111 4570.222   1.000000 71.26444  4.
## 2         South   4208.125 4011.938   1.737500 69.70625 10.
## 3 North Central   4803.000 4611.083   0.700000 71.76667  5.
## 4          West   2915.308 4702.615   1.023077 71.23462  7.
##      Frost      Area
## 1 132.7778  18141.00
## 2  64.6250  54605.12
## 3 138.8333  62652.00
```

Beispieldatensatz - apply Funktion

```
ApplyDat <- cbind(1:4,runif(4),rnorm(4))
```

```
apply(ApplyDat,1,mean)
```

```
## [1] 0.4544657 1.6193454 1.3601371 1.3806136
```

```
apply(ApplyDat,2,mean)
```

```
## [1] 2.5000000 0.4019303 0.7089911
```

Die Funktion apply

```
apply(ApplyDat,1,var)
```

```
## [1] 0.3767617 0.9290687 2.1572062 5.2211827
```

```
apply(ApplyDat,1,sd)
```

```
## [1] 0.6138092 0.9638821 1.4687431 2.2849907
```

```
apply(ApplyDat,1,range)
```

```
##             [,1]      [,2]      [,3]       [,4]
## [1,] -0.2101636 0.5232614 0.1655813 -0.2034769
## [2,]  1.0000000 2.3347749 3.0000000  4.0000000
```

```
apply(ApplyDat,1,length)
```

Argumente der Funktion apply

- Für `margin=1` die Funktion `mean` auf die Reihen angewendet,
- Für `margin=2` die Funktion `mean` auf die Spalten angewendet,
- Anstatt `mean` können auch andere Funktionen wie `var`, `sd` oder `length` verwendet werden.

Die Funktion tapply

```
ApplyDat <- data.frame(Income=rnorm(5,1400,200),
                       Sex=sample(c(1,2),5,replace=T))
```

- Auch andere Funktionen können eingesetzt werden. . . . - Auch selbst programmierte Funktionen
- Im Beispiel wird die einfachste eigene Funktion angewendet.

```
ApplyDat
```

```
##      Income Sex
## 1 1765.012   1
## 2 1430.935   1
## 3 1309.536   1
## 4 1308.268   2
## 5 1647.675   1
```

Beispiel Funktion tapply

```
tapply(ApplyDat$Income,ApplyDat$Sex,mean)
```

```
##        1        2
## 1538.289 1308.268
```

```
tapply(ApplyDat$Income,
       ApplyDat$Sex,function(x)x)
```

```
## $`1`
## [1] 1765.012 1430.935 1309.536 1647.675
##
## $`2`
## [1] 1308.268
```

Links Datenanalyse

- Benutzung von `apply`, `tapply`, etc. (R-bloggers)
- Zurück zur Gliederung