Intro Datenanalyse mit R

Jan-Philipp Kolb

05 Mai, 2019

Streuungsmaße

In Basis R sind die wichtigsten Streuungsmaße enthalten:

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

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

```
## [1] 1.073927
```

```
sd(ab)
```

```
## [1] 1.036304
```

```
range(ab)
```

```
## [1] -3.183612  2.755226
```

Extremwerte

```
min(ab)
```

```
## [1] -3.183612
```

```
max(ab)
```

```
## [1] 2.755226
```

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] 1.076646
```

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
## 12  9 13 11  2  8 13 10  8 14
```

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
## 12  9 13 11  2  8 13 10  8 14

table(x,musician)

##     musician
## x    no yes
##   1   6   6
##   2   5   4
##   3   6   7
##   4   5   6
##   5   0   2
```

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 Funktion prop.table

```
?prop.table
```

```
prop.table(table(esoph$agegp,esoph$alcgp),1)
```

```
##
##           0-39g/day     40-79    80-119      120+
##    25-34 0.2666667 0.2666667 0.2000000 0.2666667
##    35-44 0.2666667 0.2666667 0.2666667 0.2000000
##    45-54 0.2500000 0.2500000 0.2500000 0.2500000
##    55-64 0.2500000 0.2500000 0.2500000 0.2500000
##    65-74 0.2666667 0.2000000 0.2666667 0.2666667
##    75+   0.2727273 0.3636364 0.1818182 0.1818182
```

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
## 2         South   4208.125 4011.938   1.737500 69.70625
## 3 North Central   4803.000 4611.083   0.700000 71.76667
## 4          West   2915.308 4702.615   1.023077 71.23462
##       Frost       Area
## 1 132.7778  18141.00
## 2  64.6250  54605.12
## 3 138.8333  62652.00
## 4 102.1538 134463.00
```

Beispieldatensatz - apply Funktion

```
ApplyDat <- cbind(1:4,runif(4),rnorm(4))
apply(ApplyDat,1,mean)
## [1] 0.6575003 0.2212385 1.1425392 1.9401007
apply(ApplyDat,2,mean)
## [1] 2.5000000 0.1398476 0.3311864
```

Die Funktion apply

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

```
## [1] 0.1932349 3.0700083 2.5920706 3.8653434
```

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

```
## [1] 0.4395849 1.7521439 1.6099909 1.9660477
```

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

```
##           [,1]      [,2]      [,3]       [,4]
## [1,] 0.1618196 -1.503016 0.1470992 0.08374039
## [2,] 1.0000000  2.000000 3.0000000 4.00000000
```

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

```
## [1] 3 3 3 3
```

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 1721.761   2
## 2 1222.889   1
## 3 1672.644   1
## 4 1527.919   2
## 5 1511.280   1
```

Beispiel Funktion tapply

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

```
##        1        2
## 1468.937 1624.840
```

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

```
## $`1`
## [1] 1222.889 1672.644 1511.280
##
## $`2`
## [1] 1721.761 1527.919
```

Links Datenanalyse

- Die Benutzung von `apply`, `tapply`, etc. (Artikel bei R-bloggers)
- Quick-R zu deskriptiver Statistik
- Quick-R zur Funktion `aggregate`