Einführung in die Datenanalyse mit R - Datenanalyse

Jan-Philipp Kolb

8 Februar 2016

Streuungsmaße

Im base Package sind die wichtigsten Streuungsmaße enthalten:

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

```
ab <- rnorm(100)

var(ab)
```

```
## [1] 0.9607943
```

```
sd(ab)
```

```
## [1] 0.9802012
```

```
range(ab)
```

Extremwerte

```
min(ab)
```

```
## [1] -2.096978
```

```
max(ab)
```

```
## [1] 2.889542
```

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] 0.9629096
```

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
##  3 11 11  5 12 11 10  9 13 15
```

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
##  3 11 11  5 12 11 10  9 13 15
```

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

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

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

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] 0.6638817 1.5814606 1.1346008 1.9886995
```

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

```
## [1] 2.5000000 0.6399742 0.8865078
```

Die Funktion apply

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

```
## [1] 0.2120445 1.0185131 2.6309461 3.0368560
```

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

```
## [1] 0.4604829 1.0092141 1.6220191 1.7426577
```

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

```
##           [,1]      [,2]       [,3]      [,4]
## [1,] 0.1390133 0.4303142 0.05643466 0.9295828
## [2,] 1.0000000 2.3140675 3.00000000 4.0000000
```

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

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

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 1558.325   1
## 2 1169.126   2
## 3 1168.062   1
## 4 1633.222   1
## 5 1381.739   2
```

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