R-Kurs: Tag 3

Lukas Mödl, Matthias Becher, Erin Sprünken

Institut für Biometrie und Klinische Epidemiologie

Charité - Universitätsmedizin Berlin, Berlin

erin-dirk.spruenken@charite.de

January 21, 2022

Funktionen für statistische Tests

- T-Test = `t.test()`
- Chi-Quadrat-Test = `chisq.test()`
- Wilcoxon-Mann-Whitney-Test = `wilcox.test()`
- Maximum = `max()`
- Standard Deviation = `sd()`
- Variance = `var()`
- Quantile = `quantile()`
- Correlation = `cor()`
- Covariance = `cov()`
- Crosstable = `table()`

Formeln in R

Um eine Regression durchzuführen müssen wir der Funktion sagen, welche Spalten in unseren Daten die abhängigen Variablen sind und welche Spalte der abhängige Variable ist. Dafür gibt es in R die Formelschreibweise:

- Nur bestimmte Variablen sollen in der Regression verwendet werden:

$$Y \sim X1 + X2 + X3 + ...$$

- Alle Variablen sollen in der Regression verwendet werden:

$$Y \sim .$$

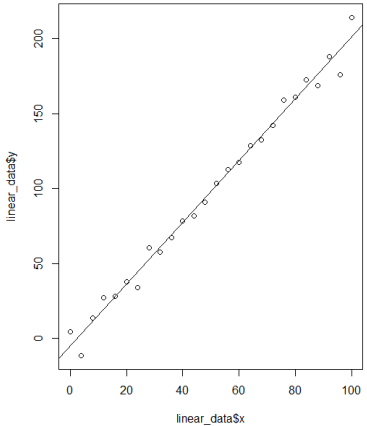
Lineare Regression

- `model <- glm(y~., data = linear_data)`
- `summary(model)`

```
Call:
glm(formula = y ~ ., data = linear_data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-16.996   -2.973    1.063    3.127   13.162

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -5.2497     2.7170  -1.932   0.0652 .
x             2.0629     0.0466  44.272   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 50.80661)

    Null deviance: 100802.2  on 25  degrees of freedom
Residual deviance:   1219.4  on 24  degrees of freedom
AIC: 179.83

Number of Fisher Scoring iterations: 2
```

Lineare Regression Plot

- `plot(linear_data$x, linear_data$y)`
- `abline(model)`



Logistische Regression

- `model <- glm(y~., data = logistic_data, family = binomial)`
- `summary(model)`

```
Call:
glm(formula = y ~ ., family = binomial, data = logistic_data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.36069  -0.27083   0.05049   0.25806   2.26779

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    6.722      1.050   6.403 1.52e-10 ***
x            -13.155      1.973  -6.666 2.63e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.24  on 199  degrees of freedom
Residual deviance: 100.78  on 198  degrees of freedom
AIC: 104.78

Number of Fisher Scoring iterations: 6
```

Apply-Familie

Die Apply-Familie ist eine Reihe von Funktion in R, die es uns erlaubt eine Funktion auf auf mehrere verschiedenen Inputs nacheinander anzuwenden. Zum Beispiel auf alle Zeilen oder Spalten einer Matrix oder alle Elemente einer Liste. Die verschiedenen Apply-Funktionen sind:

- `apply()`
- `lapply()`
- `sapply()`
- `tapply()`

apply()

Mit apply() können wir Funktionen auf alle Zeilen oder Spalten eines DataFrames oder einer Matrix anwenden um zum Beispiel alle Spaltensummen zu berechnen. Die Grundform der Funktion ist:

- `apply(data, margin, function)`

Zum Beispiel:

```
> apply(data, 1, sum)
[1] 28 32 36 40
> apply(data, 2, sum)
[1] 10 26 42 58
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

lapply()

lapply() führt eine Funktion auf jedes Element eines DataFrames, einer Matrix, eines Vektors oder einer Liste aus. Das "l" in lapply() steht dabei für "list" und bezieht sich darauf, dass lapply() immer eine Liste zurück gibt.

- `lapply(object, function)`

Zum Beispiel:

```
> lapply(c("A","B","C"), tolower)
[[1]]
[1] "a"

[[2]]
[1] "b"

[[3]]
[1] "c"
```

sapply()

Mit apply() können wir Funktionen auf alle Zeilen oder Spalten eines DataFrames oder einer Matrix anwenden um zum Beispiel alle Spaltensummen zu berechnen. Die Grundform der Funktion ist:

- `apply(data, margin, function)`

Zum Beispiel:

```
> lapply(c("A","B","C"), tolower)
[[1]]
[1] "a"

[[2]]
[1] "b"

[[3]]
[1] "c"
```

tapply()

Mit apply() können wir Funktionen auf alle Zeilen oder Spalten eines DataFrames oder einer Matrix anwenden um zum Beispiel alle Spaltensummen zu berechnen. Die Grundform der Funktion ist:

- `apply(data, margin, function)`

Zum Beispiel:

```
> apply(data, 1, sum)
[1] 28 32 36 40
> apply(data, 2, sum)
[1] 10 26 42 58
```

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$