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 28, 2022

CHARITÉ UNIVERSITÄTSMEDIZIN BERLIN

IBIKE

Statistische Tests in R

- t-Test = `t.test()`
- Chi-Quadrat Test = `chisq.test()`
- Wilcoxon-Mann-Whitney-Test = `wilcox.test()`
- Fisher Test = `fisher.test()`
- McNemar's Test = `mcnemar.test()`
- Binomial Test = `binom.test()`
- ...

t-Test

`t.test(x,...)`

Parameter:

- x = Ein Vektor mit Daten
- y = Ein optionaler Vektor mit Daten, falls man zwei Gruppen vergleichen möchte
- alternative = c("two.sided", "less", "greater")
- mu = Der angenommene Mittelwert unter der Nullhypothese
- paired = c(TRUE, FALSE)

Beispiel t-Test:

```
> t.test(rnorm(100, 0.5))

        One Sample t-test

data:  rnorm(100, 0.5)
t = 4.3709, df = 99, p-value = 3.058e-05
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 0.2449960 0.6523684
sample estimates:
mean of x
0.4486822
```

Beispiel t-Test:

```
> t.test(rnorm(200, 0.5),rnorm(200, 0.3), alternative = "greater")

        Welch Two Sample t-test

data:  rnorm(200, 0.5) and rnorm(200, 0.3)
t = 1.447, df = 397.74, p-value = 0.07434
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 -0.01962219          Inf
sample estimates:
mean of x mean of y
0.3690431 0.2282294
```

Anmerkung: Per default nimmt R beim Zwei-Stichproben-t-Test ungleiche Varianz an

Chi-Quadrat Test:

`chisq.test()`

Beispiel:

```
> table(data)
       car
color    Sedan Sportscar SUV
  black     19         18  24
  blue      24         23  17
  red       20         28  27
```

```
> chisq.test(data$color,data$car)

        Pearson's Chi-squared test

data:  data$color and data$car
X-squared = 3.5854, df = 4, p-value = 0.465
```

Formeln in R

Um eine Regression durchzuführen müssen wir der Funktion sagen, welche Spalten in unseren Daten die unabhängigen Variablen sind und welche Spalte die 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 im Datensatz sollen in der Regression verwendet werden:

$$Y \sim .$$

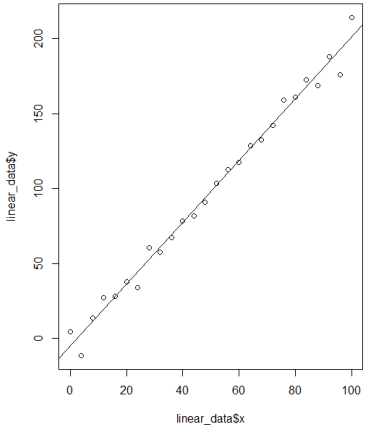
Lineare Regression

- `model <- lm(y~., data = linear_data)`
- `summary(model)`
- Anmerkung: "0 +" am Anfang der Formel führt zu einer Regression ohne Intercept

```
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
```

One-Way ANOVA

- `model <- aov(formula, data)`

```
> one.way <- aov(top_speed ~ car_type, data = data)
> summary(one.way)
             Df Sum Sq Mean Sq F value Pr(>F)
car_type      2 866583  433291    1128 <2e-16 ***
Residuals   297 114057     384
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Two-Way ANOVA

```
> two.way <- aov(top_speed~., data = data)
> summary(two.way)
             Df Sum Sq Mean Sq  F value Pr(>F)
car_type      2 866583  433291 1126.386 <2e-16 ***
color         2    578     289    0.751  0.473
Residuals   295 113479     385
```

Interaction ANOVA

```
> interaction <- aov(top_speed~ car_type*color, data = data)
> summary(interaction)
               Df Sum Sq Mean Sq  F value Pr(>F)
car_type        2 866583  433291 1123.987 <2e-16 ***
color           2    578     289    0.750  0.474
car_type:color  4   1300     325    0.843  0.499
Residuals     291 112179     385
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Apply-Familie

Die Apply-Familie ist eine Reihe von Funktion in R, die es uns erlaubt eine Funktion 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 Data Frames 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] 10 26 42 58
> apply(data, 2, sum)
[1] 28 32 36 40
```

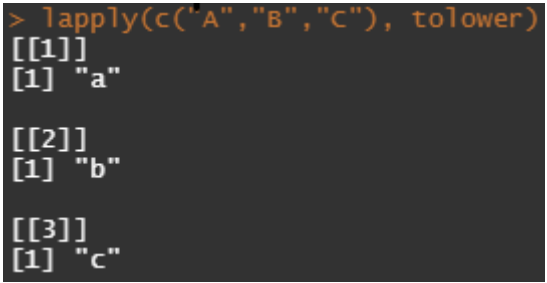
1	2	3	4	10
5	6	7	8	26
9	10	11	12	42
13	14	15	16	58
28	32	36	40	

lapply()

lapply() führt eine Funktion auf jedes Element eines Data Frames, 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:



sapply()

sapply() macht im Grunde das gleiche wie lapply(). Der Unterschied ist, dass sapply() einen Vektor oder eine Matrix zurück gibt, anstatt eine Liste:

- `sapply(object, function)`

Zum Beispiel:

```
> sapply(c(-1, 2, -3), abs)
[1] 1 2 3
> l <- list(a = 1:10, b = 1:20)
> sapply(l, mean)
    a     b
 5.5 10.5
```

tapply()

tapply() erlaubt es uns auf Grundlage von factor levels Gruppenzusammenfassungen zu erstellen:

- `tapply(object, index, function)`

Zum Beispiel:

```
> data <- data.frame(Sex = as.factor(c(rep("M",100), rep("W",100))))
> data$Height <- c(rnorm(100,180,5),rnorm(100,166,4))
> tapply(data$Height, data$Sex, mean)
       M        W
180.3354 165.3481
```