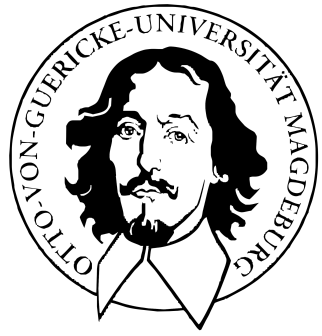


Psychotherapieforschung

MSc Klinische Psychologie und Psychotherapie

SoSe 2025

Prof. Dr. Dirk Ostwald

(2) Das Allgemeine Lineare Modell

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

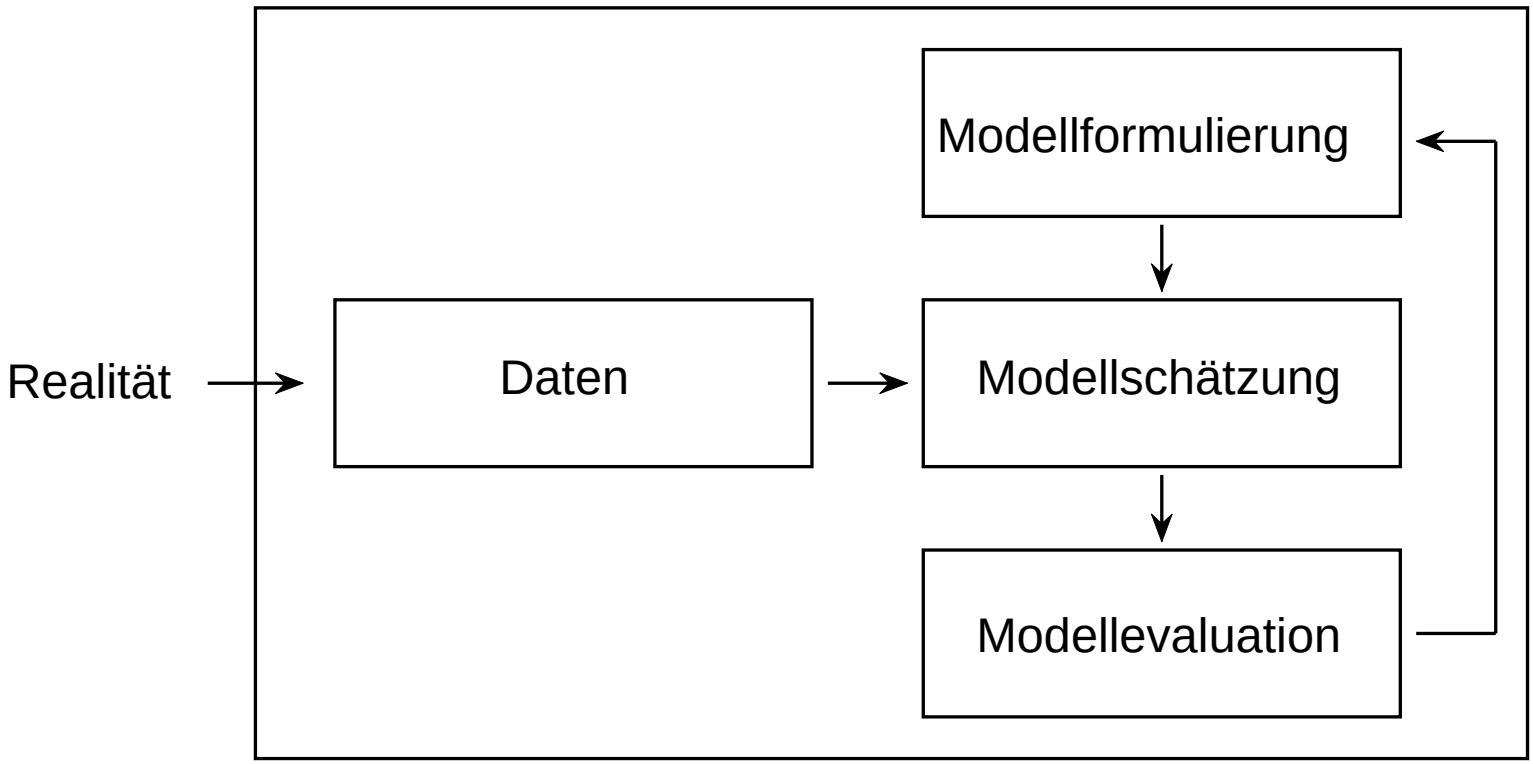
Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Naturwissenschaft als modell-basierter Realismus



Modellformulierung

$$y = X\beta + \varepsilon, \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{1}$$

Modellschätzung

$$\hat{\beta} = (X^T X)^{-1} X^T y, \quad \hat{\sigma}^2 = \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{n - p} \tag{2}$$

Modellevaluation

$$T = \frac{c^T \hat{\beta} - c^T \beta_0}{\sqrt{\hat{\sigma}^2 c^T (X^T X)^{-1} c}}, \quad F = \frac{(\hat{\varepsilon}_0^T \hat{\varepsilon}_0 - \hat{\varepsilon}^T \hat{\varepsilon})/p_1}{\hat{\varepsilon}^T \hat{\varepsilon}/(n - p)} \tag{3}$$

Definition (Allgemeines Lineares Modell)

Es sei

$$y = X\beta + \varepsilon, \tag{4}$$

wobei

- y ein n-dimensionaler beobachtbarer Zufallsvektor ist, der *Daten* genannt wird,
- $X \in \mathbb{R}^{n \times p}$ mit $n > p$ eine vorgegebene Matrix ist, die *Designmatrix* genannt wird,
- $\beta \in \mathbb{R}^p$ ein unbekannter Parametervektor ist, der *Betaparametervektor* genannt wird,
- ε ein n-dimensionaler nicht-beobachtbarer Zufallsvektor ist, der *Zufallsfehler* genannt wird und für den angenommen wird, dass mit einem unbekannten Varianzparameter $\sigma^2 > 0$ gilt, dass

$$\varepsilon \sim N\left(0_n, \sigma^2 I_n\right). \tag{5}$$

Dann heißt (4) *Allgemeines Lineares Modell (ALM)*.

Bemerkungen

- Wir nehmen durchgängig an, dass $X \in \mathbb{R}^{n \times p}$ vollen Spaltenrang hat, also dass $\text{rg}(X) = p$.
- y ist ein Zufallsvektor, weil er aus der Addition des Zufallsvektors ε zu dem Vektor $X\beta \in \mathbb{R}^n$ resultiert.
- Wir nennen $X\beta \in \mathbb{R}^n$ den *deterministischen Modellaspekt* und ε den *probabilistischen Modellaspekt*.
- $n \in \mathbb{N}$ bezeichnet durchgängig die Anzahl an Datenpunkten.
- $p \in \mathbb{N}$ bezeichnet durchgängig die Anzahl an Betaparametern.
- Die Gesamtzahl an Parametern des ALMs ist $p + 1$ (p Betaparameterkomponenten und 1 Varianzparameter).
- Der Betaparametervektor wird auch *Gewichtsvektor* oder *Effektvektor* genannt.
- Weil der Kovarianzmatrixparameter von ε als sphärisch angenommen wird, sind die $\varepsilon_1, ..., \varepsilon_n$ unabhängige normalverteilte Zufallsvariablen mit identischem Varianzparameter; weil zusätzlich der Erwartungswertparameter von ε als 0_n angenommen wird, sind die $\varepsilon_1, ..., \varepsilon_n$ auch identisch normalverteilte Zufallsvariablen.
- Für jede Komponente $y_i, i = 1, ..., n$ von y impliziert (4) nach Definition des Matrixprodukts, dass

$$y_i = x_{i1}\beta_1 + x_{i2}\beta_2 + \cdots + x_{ip}\beta_p + \varepsilon_i \text{ mit } \varepsilon_i \sim N\left(0, \sigma^2\right), \tag{6}$$

  wobei $x_{ij} \in \mathbb{R}$ das ijte Element der Designmatrix X bezeichnet.

Theorem (Datenverteilung des Allgemeinen Linearen Modells)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{7}$$

das ALM. Dann gilt

$$y \sim N(\mu, \sigma^2 I_n) \text{ mit } \mu := X\beta \in \mathbb{R}^n. \tag{8}$$

Beweis

Mit dem Theorem zur linear-affinen Transformation multivariater Normalverteilungen gilt für $\varepsilon \sim N(0_n, \sigma^2 I_n)$ und $y := I_n\varepsilon + X\beta$, dass

$$y \sim N\left(I_n 0_n + X\beta, I_n(\sigma^2 I_n)I_n^T\right) = N(X\beta, \sigma^2 I_n) = N(\mu, \sigma^2 I_n) \text{ mit } \mu := X\beta \in \mathbb{R}^n. \tag{9}$$

Bemerkungen

- Im ALM sind die Daten y also ein n-dimensionaler normalverteilter Zufallsvektor mit Erwartungswertparameter $\mu = X\beta \in \mathbb{R}^n$ und Kovarianzmatrixparameter $\sigma^2 I_n \in \mathbb{R}^{n \times n}$.
- Die Komponenten $y_1, ..., y_n$ von y, also die Datenpunkte, sind damit unabhängige, aber im Allgemeinen nicht identisch verteilte, normalverteilte Zufallsvariablen der Form $y_i \sim N\left(\mu_i, \sigma^2\right)$ für $i = 1, ..., n$.

Theorem (Betaparameterschätzer)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{10}$$

das ALM und es sei

$$\hat{\beta} := \left(X^T X\right)^{-1} X^T y. \tag{11}$$

der *Betaparameterschätzer*. Dann gilt, dass $\hat{\beta}$ die Summe der Abweichungsquadrate minimiert,

$$\hat{\beta} = \arg\min_{\tilde{\beta}} (y - X\tilde{\beta})^T (y - X\tilde{\beta}), \tag{12}$$

und dass $\hat{\beta}$ ein unverzerrter Maximum-Likelihood Schätzer von $\beta \in \mathbb{R}^p$ ist.

Bemerkungen

- Das Theorem gibt ein Formel an, um β anhand von Designmatrix und Daten zu schätzen.
- Da $\hat{\beta}$ die Summe der Abweichungsquadrate minimiert, heißt $\hat{\beta}$ auch Kleinste-Quadrate (KQ) Schätzer.
- Die $\tilde{\beta}$ Notation des Maximierungarguments dient lediglich zur Abgrenzung vom w.a.u. β.
- Als ML Schätzer ist $\hat{\beta}$ weiterhin konsistent, asymptotisch normalverteilt und asymptotisch effizient.
- Wir sehen später, dass $\hat{\beta}$ sogar normalverteilt ist.
- Außerdem hat $\hat{\beta}$ die "kleinste Varianz" in der Klasse der linearen unverzerrten Schätzer von β.
- Letztere Eigenschaft ist Kernaussage des *Gauss-Markov Theorems*, auf das wir hier nicht näher eingehen wollen.
- Für eine Diskussion und einen Beweis des Gauss-Markov Theorems siehe z.B. Searle (1971), Kapitel 3.

Theorem (Frequentistische Verteilung des Betaparameterschätzers)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{13}$$

das ALM. Weiterhin sei

$$\hat{\beta} := \left(X^T X\right)^{-1} X^T y \tag{14}$$

der Betaparameterschätzer. Dann gilt

$$\hat{\beta} \sim N\left(\beta, \sigma^2 (X^T X)^{-1}\right). \tag{15}$$

Bemerkungen

- Es gilt also wie bereits gesehen $\mathbb{E}(\hat{\beta}) = \beta$ und außerdem $\mathbb{C}(\hat{\beta}) = \sigma^2 (X^T X)^{-1}$.
- Die Varianzen der Komponenten von $\hat{\beta}$ sind die Diagonalelemente von $\mathbb{C}(\hat{\beta})$, also

$$\mathbb{V}(\hat{\beta}_i) = (\sigma^2 (X^T X)^{-1})_{ii} \text{ für } i = 1, \ldots p. \tag{16}$$

- Die Streuung von $\hat{\beta}$ hängt von σ^2 und der Designmatrix X ab. σ^2 ist ein experimentell nicht zu beinflussender wahrer, aber unbekannter, Parameter X dagegen kann so gewählt werden, um zum Beispiel die Diagonalelemente von $\mathbb{C}(\hat{\beta})$ bei festem σ^2 zu minimieren.

Theorem (Varianzparameterschätzer)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{17}$$

das ALM in generativer Form. Dann ist

$$\hat{\sigma}^2 := \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{n - p} \tag{18}$$

ein unverzerrter Schätzer von $\sigma^2 > 0$.

Bemerkungen

- Es handelt sich bei $\hat{\sigma}^2$ *nicht* um einen Maximum Likelihood Schätzer von σ^2.
- Für einen Beweis siehe z.B. Searle (1971), Kapitel 3 oder Rencher and Schaalje (2008), Kapitel 7.
- Mit Definition des Residuenvektors und der Residuen bieten sich für $\hat{\sigma}^2$ auch folgende Schreibweisen an:

$$\hat{\sigma}^2 = \frac{\hat{\varepsilon}^T \hat{\varepsilon}}{n - p} = \frac{1}{n - p} \sum_{i=1}^{n} \hat{\varepsilon}_i^2 = \frac{1}{n - p} \sum_{i=1}^{n} \left(y_i - (X\hat{\beta})_i \right)^2 \tag{19}$$

- σ^2 wird also durch die eine skalierte Residualquadratsumme geschätzt.
- Der Maximum Likelihood Schätzer des Varianzparameters ist $\hat{\sigma}^2_{\text{ML}} := \frac{1}{n} \hat{\varepsilon}^T \hat{\varepsilon}$.

Theorem (Frequentistische Verteilung des Varianzparameterschätzers)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{20}$$

das ALM. Weiterhin sei

$$\hat{\sigma}^2 = \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{n - p} \tag{21}$$

der Varianzparameterschätzer. Dann gilt

$$\frac{n-p}{\sigma^2} \hat{\sigma}^2 \sim \chi^2(n-p) \tag{22}$$

Bemerkungen

- Wir verzichten auf einen Beweis. Da es sich bei $(y-X\hat{\beta})^T(y-X\hat{\beta})$ um eine Summe quadrierter normalverteilter Zufallsvariablen handelt, liegt die χ^2-Verteilung im Lichte der χ^2 Transformation zumindest nahe.

Definition (T-Statistik)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N\left(0_n, \sigma^2 I_n\right) \tag{23}$$

das ALM. Weiterhin seien

$$\hat{\beta} := (X^T X)^{-1} X^T y \text{ und } \hat{\sigma}^2 := \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{n - p} \tag{24}$$

die Betaparameter- und Varianzparameterschätzer, respektive. Dann ist für einen *Kontrastgewichtsvektor* $c \in \mathbb{R}^p$ und einen Parameter $\beta_0 \in \mathbb{R}^p$ die *T-Statistik* definiert als

$$T := \frac{c^T \hat{\beta} - c^T \beta_0}{\sqrt{\hat{\sigma}^2 c^T (X^T X)^{-1} c}}. \tag{25}$$

Bemerkungen

- Die T-Statistik hängt via $\hat{\beta}$ und $\hat{\sigma}^2$ von den Daten y ab.
- Der Kontrastgewichtsvektor projiziert $\hat{\beta}$ auf einen Skalar $c^T \hat{\beta} \in \mathbb{R}$.
- Die Wahl p-dimensionaler Einheitsvektoren für c erlaubt die Auswahl einzelner Komponenten von $\hat{\beta}$ bzw. β_0.
- Eine generelle Wahl von c erlaubt die Evaluation beliebiger Linearkombinationen von $\hat{\beta}$ bzw. β_0.

Bemerkungen (fortgeführt)

Die Wahl von $\beta_0 \in \mathbb{R}^p$ erlaubt es, die T-Statistik unterschiedlich einzusetzen:

- Wählt man $\beta_0 := 0_p$, so erhält man mit der T-Statistik eine Deskriptivstatistik, die es erlaubt, geschätzte Regressoreffekte, also Komponenten oder Linearkombinationen von $\hat{\beta}$, im Sinne eines Signal-zu-Rauschen Verhältnisses in Bezug zu der durch $\hat{\sigma}^2$ quantifizierten Residualdatenvariabilität zu setzen. Der Nenner der T-Statistik stellt dabei sicher, dass insbesondere die adequate (Ko)Standardabweichung der entsprechenden Betaparameterkomponentenkombination als Bezugsgröße dient, da es sich bei $\hat{\sigma}^2 \left(X^T X\right)^{-1}$ bekanntlich um die Kovarianz des Betaparameterschätzers handelt. Folgende erste Intuition ist in diesem Kontext hilfreich:

$$T = \frac{\text{Geschätzte Effektstärke}}{\text{Geschätzte stichprobenumfangskalierte Datenvariabilität}} \tag{26}$$

- Wählt man für $\beta_0 = \beta$, also den wahren, aber unbekannten, Betaparameterwert, so eröffnet die T-Statistik die Möglichkeit, für einzelen Komponenten des Betaparametervektors Konfidenzintervalle zu bestimmen.
- Deklariert man schließlich $\beta_0 \in \Theta_0$ im Kontext eines Testszenarios als das Element einer Nullhypothese Θ_0, so eröffnet die T-Statistik die Hypothesentest-basierte Inferenz über Betaparameterkomponenten und ihrer Linearkombinationen.

Theorem (Frequentistische Verteilung der T-Statistik)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{27}$$

das ALM. Weiterhin seien

$$\hat{\beta} := (X^T X)^{-1} X^T y \text{ und } \hat{\sigma}^2 := \frac{(y - X\hat{\beta})^T (y - X\hat{\beta})}{n - p} \tag{28}$$

die Betaparameter- und Varianzparameterschätzer, respektive. Schließlich sei für einen Kontrastgewichtsvektor $c \in \mathbb{R}^p$ und einen Parameter $\beta_0 \in \mathbb{R}^p$

$$T := \frac{c^T \hat{\beta} - c^T \beta_0}{\sqrt{\hat{\sigma}^2 c^T (X^T X)^{-1} c}} \tag{29}$$

die T-Statistik. Dann gilt

$$T \sim t(\delta, n - p) \text{ mit } \delta := \frac{c^T \beta - c^T \beta_0}{\sqrt{\sigma^2 c^T (X^T X)^{-1} c}}. \tag{30}$$

Bemerkungen

- Wir verzichten auf einen Beweis.
- T ist eine Funktion der Parameterschätzer, δ ist eine Funktion der wahren, aber unbekannten, Parameter
- Für $c^T \beta = c^T \beta_0$, also bei Zutreffen der Nullhypothese, gilt $\delta = 0$ und damit $T \sim t(n - p)$.
- Für $c^T \beta \neq c^T \beta_0$ kann die Verteilung von T zur Herleitung von Powerfunktionen benutzt werden.

Theorem (Konfidenzintervalle für Betaparameterkomponenten)

Es sei

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \tag{31}$$

das ALM, $\hat{\beta}$ und $\hat{\sigma}^2$ seien die Betaparameter- und Varianzparameterschätzer, respektive und und für ein $\delta \in]0, 1[$ sei

$$t_\delta := \Psi^{-1}\left(\frac{1+\delta}{2}; n-p\right). \tag{32}$$

Schließlich sei für $j = 1, ...p$

$$\lambda_j := \left(\left(X^T X\right)^{-1}\right)_{jj} \text{ das } j\text{te Diagonalelement von } \left(X^T X\right)^{-1}. \tag{33}$$

Dann ist für $j = 1, ...p$

$$\kappa_j := \left[\hat{\beta}_j - \hat{\sigma}\sqrt{\lambda_j} t_\delta, \hat{\beta}_j + \hat{\sigma}\sqrt{\lambda_j} t_\delta\right] \tag{34}$$

ein δ-Konfidenzintervall für die $j$te Komponente β_j des Betaparameters $\beta = (\beta_1, ..., \beta_p)^T$.

Bemerkungen

- Intuitiv gilt im Vergleich zum Konfidenzintervall für den Erwartungswertparameter bei Normalverteilung

$$\hat{\beta}_j \approx \bar{y}, \hat{\sigma} \approx S, \sqrt{\lambda_j} \approx \sqrt{n^{-1}} \text{ und } t_\delta = t_\delta. \tag{35}$$

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

`lm()` und `formulas`

Die **R** Implementation des Allgemeinen Linearen Modells

- In R wird das ALM mit der Funktion `lm()` formuliert und geschätzt
- `lm()` wird üblicherweise in der Form `alm = lm(formula, data)` aufgerufen
- `lm()` schätzt dass durch `formula` spezifierte Modell anhand der Daten im Dataframe `data`
- Zur Modellevaluation können verschiedene Funktionen auf `alm` angewendet werden
- `summary(alm)` gibt vor allem Schätzer und Konfidenzintervalle aus
- `aov(alm)` gibt ANOVA Tabellen aus
- Eine ausführliche Dokumentation von `lm()` geben Chambers and Hastie (1992)

lm() und formulas

Modelle der Form $y = X\beta + \varepsilon$ mit $\varepsilon \sim N(0_n, \sigma^2 I_n)$ werden in R symbolisch durch `formulas` dargestellt

```
Daten ~ Term 1 + Term 2 + ... + Term k
```

- Der ~ Operator trennt die linke Seite und rechte Seite einer `formula`
- `Daten` wird zur Identifikation der abhängigen Variable y genutzt
- `Term 1 + Term 2 + ... + Term k` dient der Spezifikation der Spalten der Designmatrix X
- Die `formulas` Syntax geht zurück auf Wilkinson and Rogers (1973) und Chambers and Hastie (1992)

Terme können numerische Prädiktoren oder kategoriale Faktoren (R `factors`) sein

Die `formula` Syntax ist symbolisch, zur Laufzeit müssen die Prädiktoren und Faktoren nicht spezifiziert sein

Essentielle Operatoren in `formulas` sind + und :

- + fügt der Designmatrix Prädiktoren hinzu, : dient der Spezifikation von Interaktionen

Nichtessentielle Operatoren in `formulas` sind *, /, %in%, - und ^

- Für zwei Faktoren `f1` und `f2` gilt beispielsweise `f1*f2 = f1 + f2 + f1:f2`

lm() und formulas

Beispiele mit `f1`, `f2` als R `factors` und `x1`, `x2` als numerische Vektoren

```
formula(y ~ f1)          # Spezifikation eines einfaktoriellen ANOVA Designs mithilfe der formula() Funktion
y ~ f1                   # Aufruf der formula() Funktion ist aber nicht nötig, R erkennt formulas auch so
y ~ f1 + f2              # Additives zweifaktorielles ANOVA Design
y ~ f1 + f2 + f1:f2      # Zweifaktorielles ANOVA Design mit Interaktion
y ~ f1 + x1              # Additives einfaktorielles ANCOVA Design mit einer Kovariate

formula(y ~ x1)          # Spezifikation einer einfachen linearen Regresssion mithilfe der formula() Funktion
y ~ x1                   # Aufruf der formula() Funktion ist aber nicht nötig, R erkennt formulas auch so
y ~ 1 + x1               # Explizite Interzeptdefinition bei einfacher linearer Regression, nicht nötig
y ~ 0 + x1               # Verzicht auf Interzeptdefinition bei einfacher lineare Regression
y ~ 1 + x1 + x2          # Multiple Regression mit zwei Regressoren und expliziter Interzeptdefinition
y ~ f1 + x1 + f1:x1      # Einfaktorielles ANCOVA Design mit einer Kovariate und Interaktion
```

Wir betrachten im Folgenden die durch diese `formulas` erzeugten Designmatrizen $X \in \mathbb{R}^{n \times p}$.

lm() und formulas

Zweistichproben-T-Test = Einfaktorielles ANOVA Design mit zwei Faktorleveln

```r
n   = 12                                    # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2)) # Faktorlevel der Datenpunkte
y   = rnorm(n)                              # Beispieldaten
D   = data.frame(y = y, f1 = f1)            # Dataframe
M   = lm(y ~ f1, D)                         # Modellevaluation
X   = model.matrix(M)                       # Designmatrix
```

```
   (Intercept) f12
1            1   0
2            1   0
3            1   0
4            1   0
5            1   0
6            1   0
7            1   1
8            1   1
9            1   1
10           1   1
11           1   1
12           1   1
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

lm() und formulas

Einfaktorielles ANOVA Design mit drei Faktorleveln

```r
n   = 12                                        # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,2,2,2,2,3,3,3,3))     # Faktorlevel der Datenpunkte
y   = rnorm(n)                                  # Beispieldaten
D   = data.frame(y = y, f1 = f1)                # Dataframe
M   = lm(y ~ f1, D)                             # Modellevaluation
X   = model.matrix(M)                           # Designmatrix
```

```
   (Intercept) f12 f13
1            1   0   0
2            1   0   0
3            1   0   0
4            1   0   0
5            1   1   0
6            1   1   0
7            1   1   0
8            1   1   0
9            1   0   1
10           1   0   1
11           1   0   1
12           1   0   1
attr(,"assign")
[1] 0 1 1
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

lm() und formulas

Zweifaktorielles additives ANOVA Design mit jeweils zwei Faktorleveln

```r
n   = 12                                          # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2))       # Faktor-1-Level der Datenpunkte
f2  = as.factor(c(1,1,1,2,2,2,1,1,1,2,2,2))       # Faktor-2-Level der Datenpunkte
y   = rnorm(n)                                    # Beispieldaten
D   = data.frame(y = y, f1 = f1, f2 = f2)         # Dataframe
M   = lm(y ~ f1 + f2, D)                          # Modellevaluation
X   = model.matrix(M)                             # Designmatrix
```

```
   (Intercept) f12 f22
1            1   0   0
2            1   0   0
3            1   0   0
4            1   0   1
5            1   0   1
6            1   0   1
7            1   1   0
8            1   1   0
9            1   1   0
10           1   1   1
11           1   1   1
12           1   1   1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"

attr(,"contrasts")$f2
[1] "contr.treatment"
```

lm() und formulas

Zweifaktorielles additives ANOVA Design mit Interaktion

```r
n   = 12                                          # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2))       # Faktor-1-Level der Datenpunkte
f2  = as.factor(c(1,1,1,2,2,2,1,1,1,2,2,2))       # Faktor-2-Level der Datenpunkte
y   = rnorm(n)                                    # Beispieldaten
D   = data.frame(y = y, f1 = f1, f2 = f2)         # Dataframe
M   = lm(y ~ f1 + f2 + f1:f2, D)                  # Modellevaluation
X   = model.matrix(M)                             # Designmatrix
```

```
   (Intercept) f12 f22 f12:f22
1            1   0   0       0
2            1   0   0       0
3            1   0   0       0
4            1   0   1       0
5            1   0   1       0
6            1   0   1       0
7            1   1   0       0
8            1   1   0       0
9            1   1   0       0
10           1   1   1       1
11           1   1   1       1
12           1   1   1       1
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"

attr(,"contrasts")$f2
[1] "contr.treatment"
```

lm() und formulas

Einfache Lineare Regression

```
n   = 12                                  # Anzahl Datenpunkte
x1  = 1:n                                 # Regressor
y   = rnorm(n)                            # Beispieldaten
D   = data.frame(y = y, x1 = x1)          # Dataframe
M   = lm(y ~ x1, D)                       # Modellevaluation
X   = model.matrix(M)                     # Designmatrix
```

```
   (Intercept) x1
1            1  1
2            1  2
3            1  3
4            1  4
5            1  5
6            1  6
7            1  7
8            1  8
9            1  9
10           1 10
11           1 11
12           1 12
attr(,"assign")
[1] 0 1
```

lm() und formulas

Einfache Lineare Regression mit expliziter Interzeptdefinition

```r
n   = 12                                    # Anzahl Datenpunkte
x1  = 1:n                                   # Regressor
y   = rnorm(n)                              # Beispieldaten
D   = data.frame(y = y, x1 = x1)            # Dataframe
M   = lm(y ~ 1 + x1, D)                     # Modellevaluation
X   = model.matrix(M)                       # Designmatrix
```

```
   (Intercept) x1
1            1  1
2            1  2
3            1  3
4            1  4
5            1  5
6            1  6
7            1  7
8            1  8
9            1  9
10           1 10
11           1 11
12           1 12
attr(,"assign")
[1] 0 1
```

lm() und formulas

Einfache Lineare Regression mit Verzicht auf Interzeptdefinition

```r
n   = 12                                    # Anzahl Datenpunkte
x1  = 1:n                                   # Regressor
y   = rnorm(n)                              # Beispieldaten
D   = data.frame(y = y, x1 = x1)            # Dataframe
M   = lm(y ~ 0 + x1, D)                     # Modellevaluation
X   = model.matrix(M)                       # Designmatrix
```

```
   x1
1   1
2   2
3   3
4   4
5   5
6   6
7   7
8   8
9   9
10 10
11 11
12 12
attr(,"assign")
[1] 1
```

lm() und formulas

Multiple Regression mit zwei Regressoren und expliziter Interzeptdefinition

```
n   = 12                                  # Anzahl Datenpunkte
x1  = 1:n                                 # Regressor 1
x2  = (1:n)+5                             # Regressor 2
y   = rnorm(n)                            # Beispieldaten
D   = data.frame(y = y, x1 = x1, x2 = x2) # Dataframe
M   = lm(y ~ 1 + x1 + x2, D)              # Modellevaluation
X   = model.matrix(M)                     # Designmatrix
```

```
   (Intercept) x1 x2
1            1  1  6
2            1  2  7
3            1  3  8
4            1  4  9
5            1  5 10
6            1  6 11
7            1  7 12
8            1  8 13
9            1  9 14
10           1 10 15
11           1 11 16
12           1 12 17
attr(,"assign")
[1] 0 1 2
```

lm() und formulas

Additives einfaktorielles ANCOVA Design mit einer Kovariate

```r
n   = 12                                        # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2))     # Faktorlevel der Datenpunkte
x1  = 1:n                                       # Kovariatenwerte der Datenpunkte
y   = rnorm(n)                                  # Beispieldaten
D   = data.frame(y = y, f1 = f1, x1 = x1)       # Dataframe
M   = lm(y ~ f1 + x1, D)                        # Modellevaluation
X   = model.matrix(M)                           # Designmatrix
```

```
   (Intercept) f12 x1
1            1   0  1
2            1   0  2
3            1   0  3
4            1   0  4
5            1   0  5
6            1   0  6
7            1   1  7
8            1   1  8
9            1   1  9
10           1   1 10
11           1   1 11
12           1   1 12
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

lm() und formulas

Einfaktorielles ANCOVA Design mit einer Kovariate und Interaktion

```
n   = 12                                          # Anzahl Datenpunkte
f1  = as.factor(c(1,1,1,1,1,1,2,2,2,2,2,2))      # Faktorlevel der Datenpunkte
x1  = 1:n                                        # Kovariatenwerte der Datenpunkte
y   = rnorm(n)                                   # Beispieldaten
D   = data.frame(y = y, f1 = f1, x1 = x1)        # Dataframe
M   = lm(y ~ f1 + x1 + f1:x1, D)                 # Modellevaluation
X   = model.matrix(M)                            # Designmatrix
```

```
   (Intercept) f12 x1 f12:x1
1            1   0  1      0
2            1   0  2      0
3            1   0  3      0
4            1   0  4      0
5            1   0  5      0
6            1   0  6      0
7            1   1  7      7
8            1   1  8      8
9            1   1  9      9
10           1   1 10     10
11           1   1 11     11
12           1   1 12     12
attr(,"assign")
[1] 0 1 2 3
attr(,"contrasts")
attr(,"contrasts")$f1
[1] "contr.treatment"
```

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Definition (Kronecker-Produkt)

Es seien $A \in \mathbb{R}^{m \times n}$ und $B \in \mathbb{R}^{p \times q}$. Dann ist das *Kronecker-Produkt* von A und B definiert als die Abbildung

$$\otimes : \mathbb{R}^{m \times n} \times \mathbb{R}^{p \times q} \to \mathbb{R}^{mp \times nq},\ (A, B) \mapsto \otimes(A, B) := A \otimes B \tag{36}$$

mit

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}. \tag{37}$$

Bemerkungen

- Das Kronecker-Produkt kann zur Konstruktion von Matrizen mit repetitiver Struktur genutzt werden.
- In der Form $1_n \otimes M$ setzt das Kronecker-Produkt die Matrix M n-mal übereinander.
- In der Form $M \otimes 1_n$ repliziert das Kronecker-Produkt jede Zeile von M n-mal.
- In der Form $I_n \otimes M$ erzeugt das Kronecker-Produkt eine Blockdiagonalmatrix M auf der Diagonale.

Kronecker-Produkt

Beispiel (1)

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{pmatrix} \tag{38}$$

```
M       = matrix(c(1,1,                         # Matrix
                   1,2,
                   1,3),
                   byrow = TRUE, nrow = 3)
j_2     = matrix(rep(1,2)      , nrow = 2)      # 1_2 Vektor
X       = kronecker(j_2,M)                      # Kronecker-Produkt
```

```
     [,1] [,2]
[1,]    1    1
[2,]    1    2
[3,]    1    3
[4,]    1    1
[5,]    1    2
[6,]    1    3
```

Kronecker-Produkt

Beispiel (2)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{pmatrix} \tag{39}$$

```
M       = matrix(c(1,0,0,0,                         # Matrix
                   1,1,0,0,
                   1,0,1,0,
                   1,0,0,1),
                   byrow = TRUE, nrow = 4)
j_2     = matrix(rep(1,2)      , nrow = 2)          # 1_2 Vektor
X       = kronecker(M,j_2)                          # Kronecker-Produkt
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    1    0    0    0
[3,]    1    1    0    0
[4,]    1    1    0    0
[5,]    1    0    1    0
[6,]    1    0    1    0
[7,]    1    0    0    1
[8,]    1    0    0    1
```

Kronecker-Produkt

Beispiel (3)

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix} \tag{40}$$

```r
M       = matrix(c(2,1,                          # Matrix
                   1,2),
                   byrow = TRUE, nrow = 2)
I_2     = diag(2)                                # I_2
X       = kronecker(I_2,M)                       # Kronecker-Produkt
```

```
     [,1] [,2] [,3] [,4]
[1,]    2    1    0    0
[2,]    1    2    0    0
[3,]    0    0    2    1
[4,]    0    0    1    2
```

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Anwendungsszenario des Zweistichproben-T-Tests

- Parallelgruppendesign mit Treatmentgruppe und Kontrollgruppe
- $\Rightarrow$ Zwei Gruppen (= Stichproben) randomisierter experimenteller Einheiten.
- Annahme unabhängiger identischer Normalverteilungen $N(\mu_0, \sigma^2)$ und $N(\mu_0 + \alpha_2, \sigma^2)$.
- μ_0, α_2 und σ^2 unbekannt.
- Annahme eines identischen Varianzparameters für beide Gruppen.
- Quantifizieren der Unsicherheit beim inferentiellen Vergleich von α_2 mit 0 beabsichtigt.

Anwendungsbeispiel

- BDI-II Differenzwert-Datenanalyse bei zwei Gruppen von Patient:innen
- Treatmentfaktor (`TRM`) mit zwei Leveln (1: Waitlist Control, 2: Treatment)
- $\alpha_2 \neq 0 \Leftrightarrow$ Unterscheiden sich die wahren, aber unbekannten, Therapiewirksamkeiten?

Definition (Zweistichproben-T-Test-Modell in Effektdarstellung)

Für $i = 1, 2$ Gruppen und $j = 1, ..., n_i$ experimentelle Einheiten pro Gruppe seien y_{ij} Zufallsvariablen, die die Datenpunkte eines Zweistichproben-T-Test Szenarios modellieren. Dann hat das *Zweistichproben-T-Test Modell in Effektdarstellung* die strukturelle Form

$$\begin{aligned} y_{1j} &= \mu_0 + \varepsilon_{1j} \qquad \text{mit } \varepsilon_{1j} \sim N(0, \sigma^2) \text{ u.i.v. für } j = 1, ..., n_1 \\ y_{2j} &= \mu_0 + \alpha_2 + \varepsilon_{2j} \text{ mit } \varepsilon_{2j} \sim N(0, \sigma^2) \text{ u.i.v. für } j = 1, ..., n_2 \end{aligned} \tag{41}$$

und die entsprechende Datenverteilungsform

$$\begin{aligned} y_{1j} &\sim N(\mu_0, \sigma^2) \qquad \text{u.i.v. für } j = 1, ..., n_i \text{ mit } \mu_0 \in \mathbb{R}, \sigma^2 > 0 \\ y_{2j} &\sim N(\mu_0 + \alpha_2, \sigma^2) \text{ u.i.v für } j = 1, ..., n_2 \text{ mit } \mu_0, \alpha_2 \in \mathbb{R}, \sigma^2 > 0 \end{aligned} \tag{42}$$

Bemerkung

- Das Zweistichproben-T-Testmodell in Effektdarstellung entspricht dem einfaktoriellen Varianzanalysemodell in Effektdarstellung mit Referenzgruppe für einen experimentellen Faktor mit zwei Leveln.

Theorem (Zweistichproben-T-Test-Modell in Effektdarstellung)

Gegeben sei die strukturelle Form des Zweistichproben-T-Test Modells in Effektdarstellung. Dann hat dieses Modell die Designmatrixform

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n), n := n_1 + n_2 \tag{43}$$

und

$$y := \begin{pmatrix} y_{11} \\ \vdots \\ y_{1n_1} \\ y_{21} \\ \vdots \\ y_{2n_2} \end{pmatrix} \in \mathbb{R}^n,\ X := \begin{pmatrix} 1 & 0 \\ \vdots & \vdots \\ 1 & 0 \\ 1 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{pmatrix} \in \mathbb{R}^{n \times 2},\ \beta := \begin{pmatrix} \mu_0 \\ \alpha_2 \end{pmatrix} \in \mathbb{R}^2 \text{ und } \sigma^2 > 0. \tag{44}$$

Bemerkung

- Das Theorem ergibt sich direkt mit den Regeln der Matrixmultiplikation.

Datengeneration

```r
library(MASS)                                                  # multivariate Normalverteilung
set.seed(0)                                                    # Zufallszahlengeneratorzustand
n_j     = 2                                                    # Anzahl Bedingungen
n_i     = 10                                                   # Patient:innen pro  Bedingung
n       = n_i*n_j                                              # Gesamtanzahl an Patient:innen
X       = kronecker(matrix(c(1,1,0,1), ncol = 2), rep(1,n_i))  # Treatment-Control Designmatrix
beta    = matrix(c(5,2), nrow = n_j)                           # Fixed-Effects-Parameter
s_eps   = 1                                                    # Varianzparameter
eps     = mvrnorm(1, rep(0,n), s_eps*diag(n))                  # Fehlervektor
y       = X %*% beta + eps                                     # Datengeneration
TRM     = kronecker(c(1:n_j), rep(1,n_i))                      # Treatmentfaktor
D       = data.frame(TRM = TRM, BDI =  y)                      # Dataframe
write.csv(D, "./2_Daten/ztt.csv", row.names = FALSE)           # Speichern
```

Datensatz

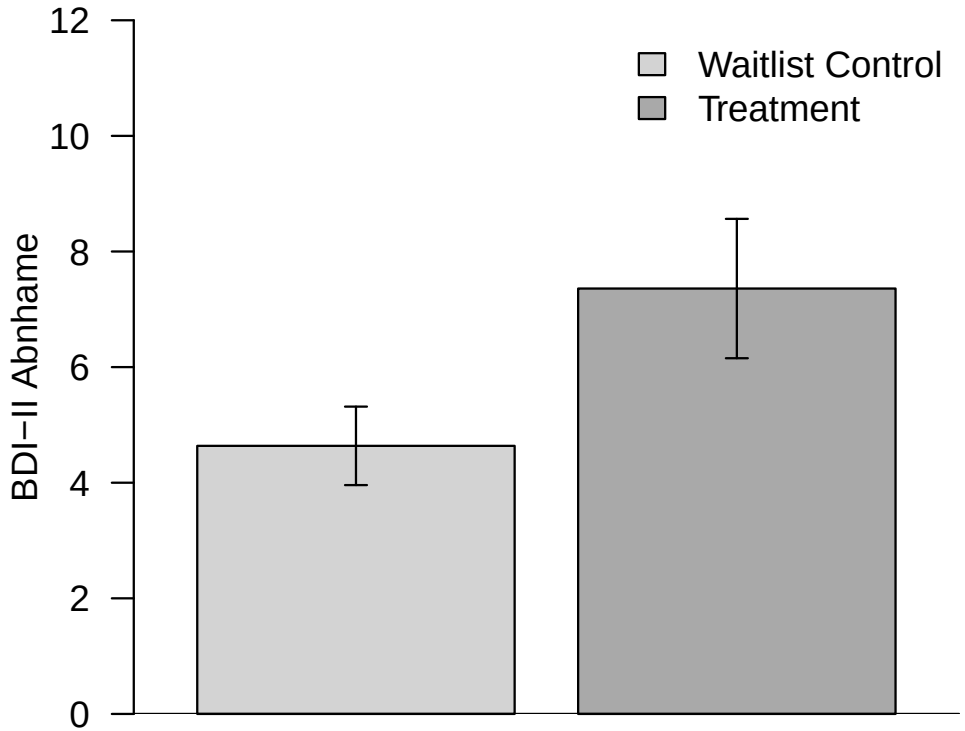
TRM	BDI
1	3.8
1	5.4
1	4.1
1	5.3
1	4.6
1	4.7
1	4.7
1	3.9
1	4.2
1	5.8
2	9.4
2	7.0
2	6.7
2	6.1
2	5.5
2	7.4
2	8.3
2	8.3
2	6.7
2	8.3

Deskriptivstatistik

```
library(dplyr)                                                    # dplyr für einfache Datengruppierug
D   = read.csv("./2_Daten/ztt.csv")                               # Dateneinlesen
DS  = D %>% group_by(TRM) %>%  summarise(av = mean(BDI, na.rm = TRUE), # Group mean
                                         sd = sd(BDI, na.rm = TRUE),   # Group standard deviation
                                         .groups = "drop")             # Gruppierungsaufhebung
print(DS)
```

```
# A tibble: 2 x 3
    TRM    av    sd
  <int> <dbl> <dbl>
1     1  4.64 0.679
2     2  7.36 1.21
```

Visualisierung



Modellschätzung und Modellevaluation

```r
D           = read.csv("./2_Daten/ztt.csv")                    # Dateneinlesen
n_j         = 2                                                # Anzahl Bedingungen
n_i         = 10                                               # Patient:innen pro Bedingung
n           = n_i*n_j                                          # Gesamtanzahl an Patient:innen
p           = 2                                                # Anzahl Betaparameter
X           = kronecker(matrix(c(1,1,0,1), ncol = p), rep(1,n_i)) # Treatment-Control Designmatrix
y           = D$BDI                                            # Daten
beta_hat    = solve(t(X) %*% X) %*% t(X) %*% y                 # Betaparameterschätzer
eps_hat     = y - X %*% beta_hat                               # Prädiktionsfehler
sigsqr_hat  = (t(eps_hat) %*% eps_hat)/(n-p)                   # Varianzparameterschätezr
delta       = 0.95                                             # Konfidenzbedingung
t_delta     = qt((1+delta)/2,n-p)                              # \Psi^{-1}((1+\delta)/2,n-1)
lambda      = diag(solve(t(X) %*% X))                          # \lambda_j Werte
ses         = sqrt(sigsqr_hat)*sqrt(lambda)                    # Betaparameterstandardfehlerschätzer
tvals       = beta_hat/ses                                     # T Werte
kappa_u     = beta_hat - ses*t_delta                           # untere KI Grenze
kappa_o     = beta_hat + ses*t_delta                           # obere  KI Grenze
```

Modellschätzung und Modellevaluation

```
# Ergebnisausgabe
print(
data.frame(
"Estimate"   = beta_hat,
"Std.Error"  = ses,
"t.value"    = beta_hat/ses,
"p.value"    = 2*(1-pt(abs(tvals), n-p)),
"KI.u"       = kappa_u,
"KI.o"       = kappa_o,
row.names    = c("mu_0_hat", "alpha_2_hat")),
digits = 4)
```

```
            Estimate Std.Error t.value   p.value  KI.u  KI.o
mu_0_hat       4.638    0.3093  14.992 1.300e-11 3.988 5.287
alpha_2_hat    2.721    0.4375   6.221 7.191e-06 1.802 3.640
```

Modellschätzung und Modellevaluation mit `lm()`

```
D           = read.csv("./2_Daten/ztt.csv", head = T)          # Dataframe
D$TRM       = as.factor(D$TRM)                                 # R Faktor Kodierung
M           = lm(BDI ~ TRM, data = D)                          # ALM Schätzung
coefs       = summary(M)$coefficients                          # Betaparameterschätzer
ci          = confint(M, level  = 0.95)                        # Konfidenzintervalle
print(cbind(coefs, KI.u = ci[, 1], KI.o = ci[, 2]), digits = 4)
```

```
            Estimate Std. Error t value  Pr(>|t|)  KI.u  KI.o
(Intercept)    4.638     0.3093  14.992 1.300e-11 3.988 5.287
TRM2           2.721     0.4375   6.221 7.191e-06 1.802 3.640
```

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Anwendungsszenario der einfachen linearen Regression

- Frage nach dem linear-affinen Zusammenhang univariater UV und AV
- Annahme einer linear-affinen funktionalen Abhängigkeit der Form

$$y = \beta_0 + \beta_1 x + \varepsilon \text{ mit } \varepsilon \sim N(0, \sigma^2) \tag{45}$$

- β_0: Schnittpunkt von Gerade und y-Achse ("Intercept Parameter")
- β_1: y-Differenz pro x-Einheitsdifferenz ("Slope Parameter")
- β_0, β_1 und σ^2 unbekannt.
- Quantifizieren der Unsicherheit bei Parameterinferenz beabsichtigt.

Anwendungsbeispiel

- Dosis-Wirkungs-Kurvenuntersuchung bei Psychotherapie
- x: Anzahl Therapiestunden (`ATS`) y: BDI-II Abnahme (`BDI`)
- $\beta_1 < 0$ für einen linearen Zusammenhang

Definition (Modell der einfachen linearen Regression I)

Für $i = 1, \ldots, n$ experimentelle Einheiten hat das *Modell der einfachen linearen Regression* die strukturelle Form

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \text{ mit } \varepsilon_i \sim N(0, \sigma^2) \tag{46}$$

wobei

- y_i beobachtbare Zufallsvariablen sind, die Werte einer abhängigen Variable modellieren,
- $x_i \in \mathbb{R}$ fest vorgegebene sogenannte *Prädiktorwerte* oder *Regressorwerte* sind, die Werte einer unabhängigen Variable modellieren
- $\beta_0, \beta_1 \in \mathbb{R}$ wahre, aber unbekannte, Intercept- und Slope-Parameterwerte sind und
- $\varepsilon_i \sim N(0, \sigma^2)$ unabhängig und identisch normalverteilte nicht-beobachtbare Zufallsvariablen mit wahrem, aber unbekanntem, Varianzparameter $\sigma^2 > 0$ sind, die Messfehler modellieren.

und die entsprechende Datenverteilungsform

$$y_i \sim N\left(\mu_i, \sigma^2\right) \text{ u.v. für } \text{ mit } \mu_i := \beta_0 + \beta_1 x_i \text{ und } i = 1, \ldots, n \tag{47}$$

Bemerkung

- Die Werte der abhängigen Variable werden im Modell der einfachen linearen Regression also durch unabhängige normalverteilte Zufallsvariablen mit im Allgemeinen unterschiedlichen Erwartungswertparametern modelliert.

Theorem (Modell der einfachen linearen Regression II)

Gegeben sei die strukturelle Form des Modells der einfachen linearen Regression Dann hat dieses Modell die Designmatrixform

$$y = X\beta + \varepsilon \text{ mit } \varepsilon \sim N(0_n, \sigma^2 I_n) \text{ für } n := n_1 + n_2 \tag{48}$$

und

$$y := \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n,\ X := \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \in \mathbb{R}^{n \times 2},\ \beta := \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \in \mathbb{R}^2 \text{ und } \sigma^2 > 0. \tag{49}$$

Bemerkung

- Das Theorem ergibt sich direkt mit den Regeln der Matrixmultiplikation.

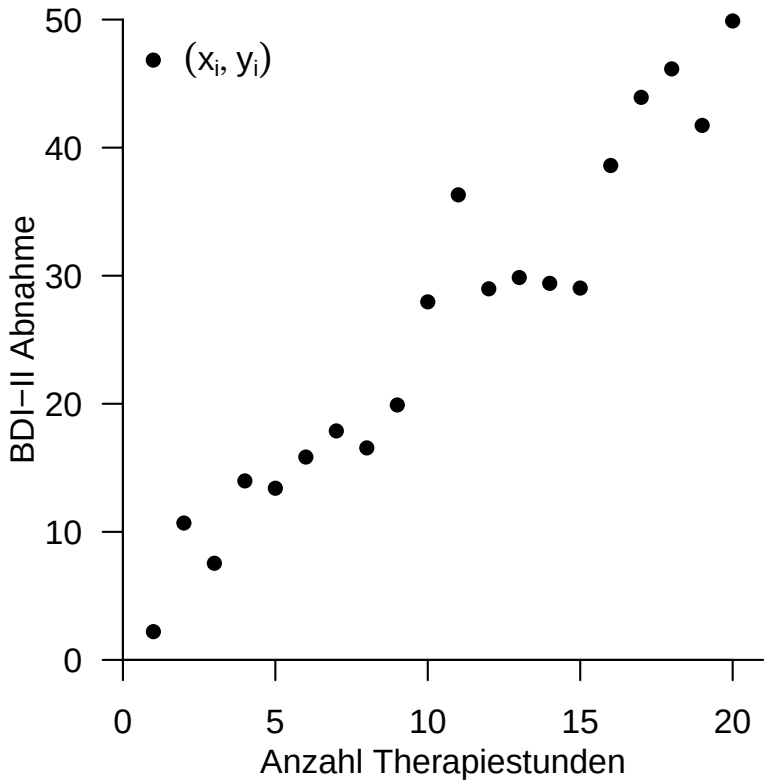
Datengeneration

```r
library(MASS)                                          # multivariate Normalverteilung
set.seed(0)                                            # Zufallszahlengeneratorzustand
n       = 20                                           # Gesamtanzahl an Patient:innen
X       = matrix(c(rep(1,n), 1:n), ncol = 2)           # Designmatrix
beta    = matrix(c(5,2), nrow = 2)                     # Betaparameter
s_eps   = 15                                           # Varianzparameter
eps     = mvrnorm(1, rep(0,n), s_eps*diag(n))          # Fehlervektor
y       = X %*% beta + eps                             # Datengeneration
D       = data.frame(ATS = X[,2], BDI =  y)            # Dataframe
write.csv(D, "./2_Daten/elr.csv", row.names = FALSE)   # Speichern
```

Datensatz

ATS	BDI
1	2.2
2	10.7
3	7.5
4	14.0
5	13.4
6	15.8
7	17.9
8	16.6
9	19.9
10	28.0
11	36.3
12	29.0
13	29.9
14	29.4
15	29.0
16	38.6
17	43.9
18	46.2
19	41.7
20	49.9

Visualisierung



Modellschätzung und Modellevaluation

```r
D           = read.csv("./2_Daten/elr.csv")                  # Dateneinlesen
y           = D$BDI                                          # Daten
X           = matrix(c(rep(1,n), D$ATS), ncol = 2)           # Designmatrix
n           = nrow(X)                                        # Gesamtanzahl an Patient:innen
p           = ncol(X)                                        # Anzahl Betaparameter
beta_hat    = solve(t(X) %*% X) %*% t(X) %*% y               # Betaparameterschätzer
eps_hat     = y - X %*% beta_hat                             # Prädiktionsfehler
sigsqr_hat  = (t(eps_hat) %*% eps_hat)/(n-p)                 # Varianzparameterschätezr
delta       = 0.95                                           # Konfidenzbedingung
t_delta     = qt((1+delta)/2,n-p)                            # \Psi^{-1}((1+\delta)/2,n-1)
lambda      = diag(solve(t(X) %*% X))                        # \lambda_j Werte
ses         = sqrt(sigsqr_hat)*sqrt(lambda)                  # Betaparameterstandardfehlerschätzer
tvals       = beta_hat/ses                                   # T Werte
kappa_u     = beta_hat - ses*t_delta                         # untere KI Grenze
kappa_o     = beta_hat + ses*t_delta                         # obere  KI Grenze
```

Modellschätzung und Modellevaluation

```
# Ergebnisausgabe
print(
data.frame(
"Estimate"   = beta_hat,
"Std.Error"  = ses,
"t.value"    = beta_hat/ses,
"p.value"    = 2*(1-pt(abs(tvals), n-p)),
"KI.u"       = kappa_u,
"KI.o"       = kappa_o,
row.names    = c("beta_0_hat", "beta_1_hat")),
digits = 4)
```

```
           Estimate Std.Error t.value   p.value   KI.u  KI.o
beta_0_hat    2.459     1.761   1.396 1.796e-01 -1.240 6.159
beta_1_hat    2.241     0.147  15.248 9.799e-12  1.933 2.550
```

Modellschätzung und Modellevaluation mit `lm()`

```
D           = read.csv("./2_Daten/elr.csv", head = T)           # Dataframe
M           = lm(BDI ~ ATS, data = D)                           # ALM Schätzung
coefs       = summary(M)$coefficients                           # Betaparameterschätzer
ci          = confint(M, level  = 0.95)                         # Konfidenzintervalle
print(cbind(coefs, KI.u = ci[, 1], KI.o = ci[, 2]), digits = 4)
```

```
            Estimate Std. Error t value  Pr(>|t|)   KI.u  KI.o
(Intercept)    2.459      1.761   1.396 1.796e-01 -1.240 6.159
ATS            2.241      0.147  15.248 9.799e-12  1.933 2.550
```

Modellformulierung, Modellschätzung, Modellevaluation

`lm()` und `formulas`

Kronecker-Produkt

Zweistichproben-T-Test

Einfache lineare Regression

Selbstkontrollfragen

Selbstkontrollfragen

1. Geben Sie die Definition des Allgemeinen Linearen Modells wieder und erläutern Sie sie.
2. Geben Sie das Theorem zur Datenverteilung des Allgemeinen Linearen Modells wieder.
3. Geben Sie das Theorem zum Betaparameterschätzer wieder und erläutern Sie es.
4. Geben Sie das Theorem zur Frequentistischen Verteilung des Betaparameterschätzers wieder.
5. Geben Sie das Theorem zum Varianzparameterschätzer wieder und erläutern Sie es.
6. Geben Sie das Theorem zur Frequentistischen Verteilung des Varianzparameterschätzers wieder.
7. Geben Sie die Definition der T-Statistik wieder und erläutern Sie sie in Abhängigkeit von β_0.
8. Geben Sie das Theorem zur Frequentistischen Verteilung der T-Statistik wieder.
9. Geben Sie das Theorem zu den Konfidenzintervallen für Betaparameterkomponenten wieder.
10. Erläutern Sie die Syntax und Semantik der R `lm()` Funktion.
11. Erläutern Sie die Syntax und Semantik von R `formulas`.
12. Geben Sie die Definition des Kronecker-Produkts zweier Matrizen wieder.
13. Erläutern Sie das Anwendungsszenario eines Zweistichproben-T-Tests.
14. Geben Sie die strukturelle Form, die Datenverteilungsform und die Designmatrixform des Zweistichproben-T-Testmodells in Effektdarstellung wieder.
15. Erläutern Sie das Anwendungsszenario einer einfachen linearen Regression.
16. Geben Sie die strukturelle Form, die Datenverteilungsform und die Designmatrixform des Modells der einfachen linearen Regression wieder.

Referenzen

Chambers, John M., and Trevor Hastie, eds. 1992. *Statistical Models in S*. Wadsworth & Brooks/Cole Computer Science Series. Pacific Grove, Calif: Wadsworth & Brooks/Cole Advanced Books & Software.

Rencher, Alvin C., and G. Bruce Schaalje. 2008. *Linear Models in Statistics*. 2nd ed. Hoboken, N.J: Wiley-Interscience.

Searle, S. R. 1971. *Linear Models*. Wiley Classics Library. New York, NY: Wiley.

Wilkinson, G. N., and C. E. Rogers. 1973. "Symbolic Description of Factorial Models for Analysis of Variance." *Applied Statistics* 22 (3): 392. https://doi.org/10.2307/2346786.