# Evaluation und Metaanalyse

MSc Klinische Psychologie und Psychotherapie

SoSe 2024

Prof. Dr. Dirk Ostwald

# (7) Linear Mixed Models in R

Linear Models mit `lm()` und `gls()`

Linear Mixed Models mit `nlme`

Linear Mixed Models mit `lme4.0`

Selbstkontrollfragen

**Linear Models mit `lm()` und `gls()`**

Linear Mixed Models mit `nlme`

Linear Mixed Models mit `lme4.0`

Selbstkontrollfragen

# Linear Models mit `lm()` und `gls()`

## R 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) (vgl. 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`

# Linear Models mit `lm()` und `gls()`

## R formulas

Beispiele mit `x1` als numerischer Vektor und `f1`, `f2` als R Faktoren

```
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 ~ f1                   # Einfaktorielles ANOVA Design
y ~ f1 + f2              # Additives zweifaktorielles ANOVA Design
y ~ f1 + f2 + f1:f2      # Zweifaktorielles ANOVA Design mit Interaktion
y ~ f1*f2                # Zweifaktorielles ANOVA Design mit Interaktion
y ~ f1 + x1              # Additives einfaktorielles ANCOVA Design mit einer Kovariate
y ~ f1 + x1 + f1:x1      # Einfaktorielles ANCOVA Design mit einer Kovariate und Interaktionp
```

Wir betrachten die durch diese `formulas` erzeugten Designmatrizen im Folgenden

# Linear Models mit `lm()` und `gls()`

## Die `terms` Klasse

Die `terms()` Konstruktionsfunktion wandelt `formulas` in Objekte zur Konstruktion von Designmatrizen um

Wir betrachten das Beispiel eines Zweifaktoriellen ANOVA Designs mit Interaktion

```
t   = terms(y ~ f1*f2)       # Anwendung der terms() Konstruktionsfunktion auf eine formula
names(attributes(t))         # Attributnamen des terms Objekts
```

```
[1] "variables"    "factors"      "term.labels"  "order"        "intercept"
[6] "response"     "class"        ".Environment"
```

```
labels(t)                    # Terme der formula
```

```
[1] "f1"    "f2"    "f1:f2"
```

```
attr(t, "order")             # Interaktionsordnung der Terme
```

```
[1] 1 1 2
```

```
attr(t, "intercept")         # Interzeptinklusion
```

```
[1] 1
```

```
attr(t, "variables")         # Variablennamen
```

```
list(y, f1, f2)
```

## Die `model.frame()` Funktion

Die `model.frame()` Funktion kombiniert ein `terms` mit einem `data.frame` Objekt

Ein `model.frame` Objekt dient als Präkursor einer Designmatrix.

Die `model.matrix()` Funktion

Linear Models mit `lm()` und `gls()`

**Linear Mixed Models mit `nlme`**

Linear Mixed Models mit `lme4.0`

Selbstkontrollfragen

# Linear Mixed Models mit `nlme`

## Die `pdMat` Klasse

Konstruktion positiv-definiter Matrizen für die Random-Effects-Kovarianzmatrix

```
library(nlme)
?pdClasses
```

| Funktion | Zweck |
|---|---|
| `pdIdenit` | Konstruktion sphärischer Kovarianzmatrizen |
| `pdDiag` | Konstruktion von Diagonalkovarianzmatrizen |
| `pdCompSymm` | Konstruktion von Kovarianzmatrizen mit Compound Symmetry |
| `pdLogChol` | Konstruktion von Kovarianzmatrizen mithilfe der Log-Cholesky-Parameterisierung |
| `pdSymm` | Konstruktion von Kovarianzmatrizen mithilfe einer SVD-Parameterisierung |
| `pdNatural` | Konstruktion von Kovarianzmatrizen mithilfe von Standardabweichungen und Korrelationen |
| `pdBlocked` | Konstruktion von Blockdiagonalkovarianzmatrizen mithilfe obiger Funktionen |

Generelle Argumente der Konstruktionsfunktionen sind `value`, `form`, `data` und `nam`

- `value` erlaubt die Spezifikation mithilfe selbst gewählter Werte
- `form` und `data` erlauben eine R `formula` basierte Konstruktion
- `nam` wird zur Benennung von Zeilen und Spalten der Kovarianzmatrix genutzt

`methods(class = "pdMat")` gibt Funktionen zur Inspektion und Manipulation von `pdMat` Objekten an

- `summary()` gibt einen Überblick

## Die pdMat Klasse

```
# Konstruktion einer sphärischen Kovarianzmatrix
library(nlme)                                              # nlme Paket
n   = 5                                                    # Dimension
I   = pdIdent(diag(n))                                     # value-basierte Spezifikation
pdMatrix(I)                                                # konstruierte Matrix
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

```
Dim(I)                                                     # Dimension der Matrix
```

```
[1] 5 5
```

```
summary(I)                                                 # Zusammenfassung
```

```
 Structure: Multiple of an Identity
        V1 V2 V3 V4 V5
StdDev:  1  1  1  1  1
```

## Die pdMat Klasse

```
# Konstruktion einer Diagonalkovarianzmatrix
library(nlme)                                           # nlme Paket
D    = pdDiag(diag(1:5))                                # value-basierte Spezifikation
pdMatrix(D)                                             # konstruierte Matrix
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    2    0    0    0
[3,]    0    0    3    0    0
[4,]    0    0    0    4    0
[5,]    0    0    0    0    5
```

```
Dim(D)                                                  # Dimension der Matrix
```

```
[1] 5 5
```

```
summary(D)                                              # Zusammenfassung
```

```
 Structure: Diagonal
        V1       V2       V3 V4       V5
StdDev:  1 1.414214 1.732051  2 2.236068
```

Die `reStruct` Klasse

Spezifikation der Random-Effects-Designmatrix und der Random-Effects-Kovarianzmatrix

Die `lmeStruct` Klasse

Spezifikation der Random-Effects Aspekte eines Linear Mixed Models `lme()` für Modellformulierung und Modellschätzung

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.

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.