AS17: Kapitel 8 - Modellbildung

Jürgen Hedderich

2020-04-28

Contents

Hinweis: Die Gliederung zu den Befehlen Und Ergebnissen in R orientiert sich an dem Inhaltsverzeichnis der 17. Auflage der 'Angewandten Statistik'. Nähere Hinweise zu den verwendeten Formeln sowie Erklärungen zu den Beispielen sind in dem Buch (Ebook) nachzulesen!

zur Druckversion

Hinweis: Die thematische Gliederung zu den aufgeführten Befehlen und Beispielen orientiert sich an dem Inhaltsverzeichnis der 17. Auflage der 'Angewandten Statistik'. Nähere Hinweise zu den verwendeten Formeln sowie Erklärungen zu den Beispielen sind in dem Buch (Ebook) nachzulesen!

Aktuelle R-Umgebung zu den Beispielen

The R Project for Statistical Computing

```
sessionInfo()
## R version 3.6.3 (2020-02-29)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18363)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.6.3  magrittr_1.5    tools_3.6.3     htmltools_0.4.0
##  [5] yaml_2.2.1      Rcpp_1.0.4      stringi_1.4.6   rmarkdown_2.1
##  [9] knitr_1.28      stringr_1.4.0   xfun_0.12       digest_0.6.25
## [13] rlang_0.4.5     evaluate_0.14
```

Download von Datensätzen, die in den folgenden Beispielen verwendet werden:

Cholesterin: cholesterin

Hemmhof: hemmhof

Hemmhof-2: hemmhof2

Propensity-Score: psdata

Hautkrebs: skincancer

Messwiederholungen: lmebroad

Cluster: cluster

Teratogenität: teratogen

Chemotherapie: chemotherapie

Sterbetafel: sterbetafel

8.1 Einführung

Bootstrapping:

```
x    <- rnorm(50, mean=10, sd=5)

stat <- function(x) quantile(x, probs=0.25); stat(x)
##      25%
## 6.157745

B    <- 1000; t <- rep(NA, B)
for (i in 1:B) t[i] <- stat(sample(x, replace=TRUE))

E.stat <- mean(t); E.stat                                    # Erwartungswert
## [1] 6.087485

V.stat <- var(t); V.stat                                     # Varianz
## [1] 0.534805
```

Jackknife:

```
x     <- rnorm(50, mean=10, sd=5)

theta <- function(x) quantile(x, probs=0.95); theta(x)
##      95%
## 18.05744

n     <- length(x)
t     <- rep(NA, n)
for  (i in 1:n) t[i] = theta(x[-i])

JE.theta <- mean(t); JE.theta                                # Schätzung
## [1] 18.03536

JV.theta <- ((n-1)/n) * sum((t-JE.theta)^2); JV.theta        # Varianz
## [1] 0.705413

JC.theta <- n*theta(x) - (n-1)*JE.theta; JC.theta            # Korrektur
##      95%
## 19.13918
```

Cross validation:

```r
x <- seq(0, 10, by = 0.5)

n <- length(x)                                          # lineares Regressionsmodell
y <- 2 + 0.5*x; y <- y + rnorm(n, mean=0, sd=0.6)
resi <- lm(y~x)$residuals; sum(resi^2)                  # Residuen
## [1] 8.378676

err <- rep(NA, n)
for (i in 1:n) {
    mcf    <- lm(y[-i] ~ x[-i])$coef                    # Koeffizienten
    y.hat  <- mcf[1] + mcf[2]*x[i]                      # Schätzung
    err[i] <- y[i] - y.hat  }                           # Schätz-Fehler

t.err <- sum(err^2); t.err
## [1] 10.25057
```

8.2 Lineare Regressionsmodelle

8.2.1 Die einfache lineare Regression

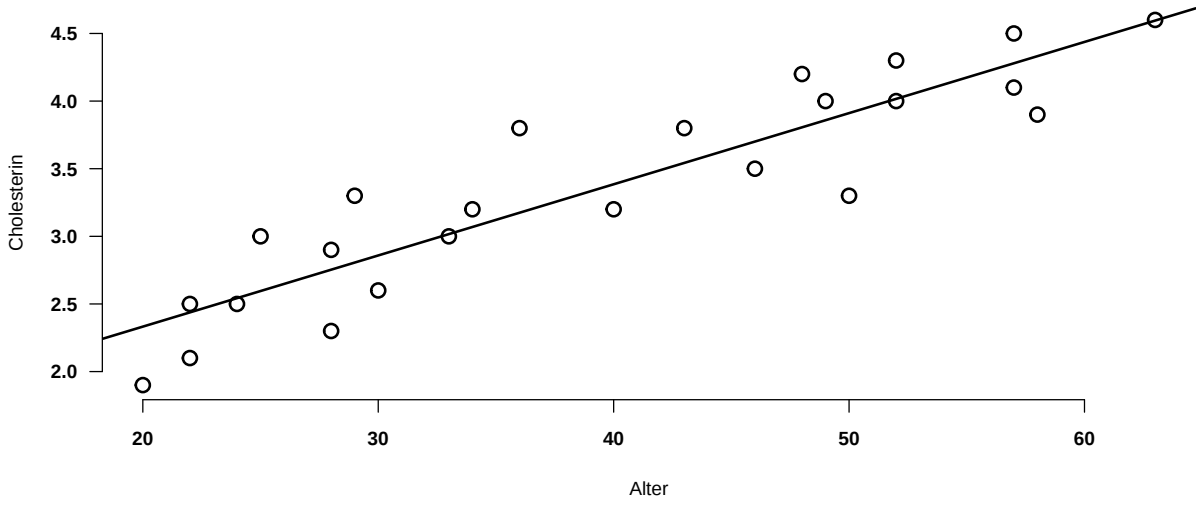
Beispiel Cholesterin und Alter:

```
cholesterin <- read.csv2("cholesterin.CSV", sep=";",dec=",",header=T)
as.data.frame(cholesterin[1:10,])
```

Id	Alter	Chol
a	46	3.5
b	20	1.9
c	52	4.0
d	30	2.6
e	57	4.5
f	25	3.0
g	28	2.9
h	36	3.8
i	22	2.1
j	43	3.8

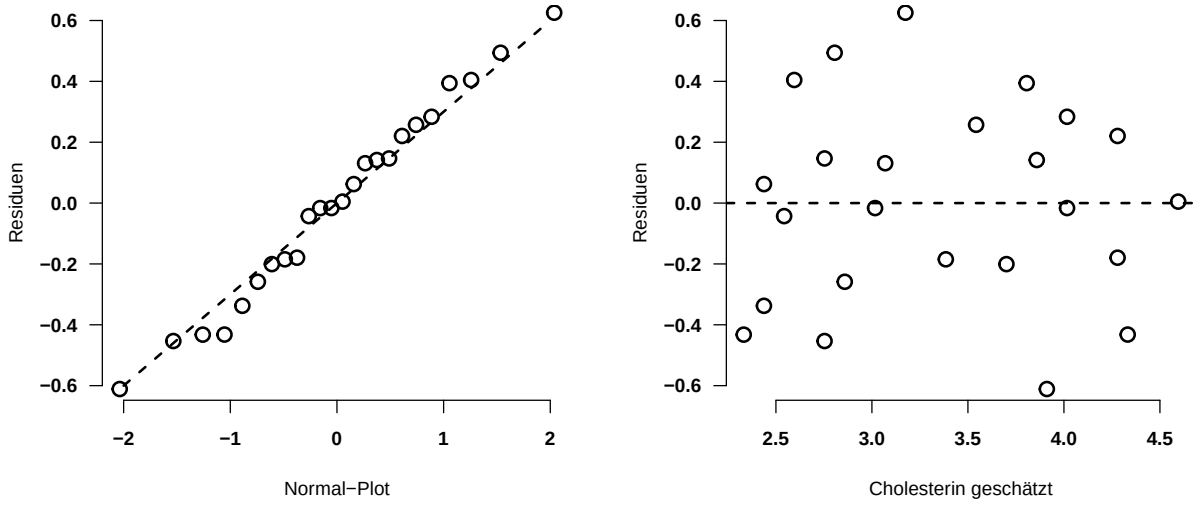
```
attach(cholesterin)

n    <- length(Alter)                                    # Maßzahlen
m.x  <- mean(Alter);                  m.x
## [1] 39.41667
s.x  <- sd(Alter);                    s.x
## [1] 13.41614
m.y  <- mean(Chol);                   m.y
## [1] 3.354167
s.y  <- sd(Chol);                     s.y
## [1] 0.7779455
                                                         # Varianz- und Kovarianz
ss.xx <- sum((Alter - m.x)^2);             ss.xx
## [1] 4139.833
ss.yy <- sum((Chol  - m.y)^2);             ss.yy
## [1] 13.91958
ss.xy <- sum((Alter - m.x)*(Chol - m.y));  ss.xy
## [1] 217.8583
                                                         # Regressionskoeffizienten
beta1 <- ss.xy / ss.xx;                    beta1
## [1] 0.0526249
beta0 <- m.y - beta1 * m.x;                beta0
## [1] 1.279868
```

```
par(lwd=2, font.axis=2, bty="n", ps=11, mfrow=c(1,1))
plot(Alter, Chol, ylab="Cholesterin", las=1, cex=1.5)        # Punktwolke
abline(beta0, beta1, col="black")                            # Regressionsgerade
```

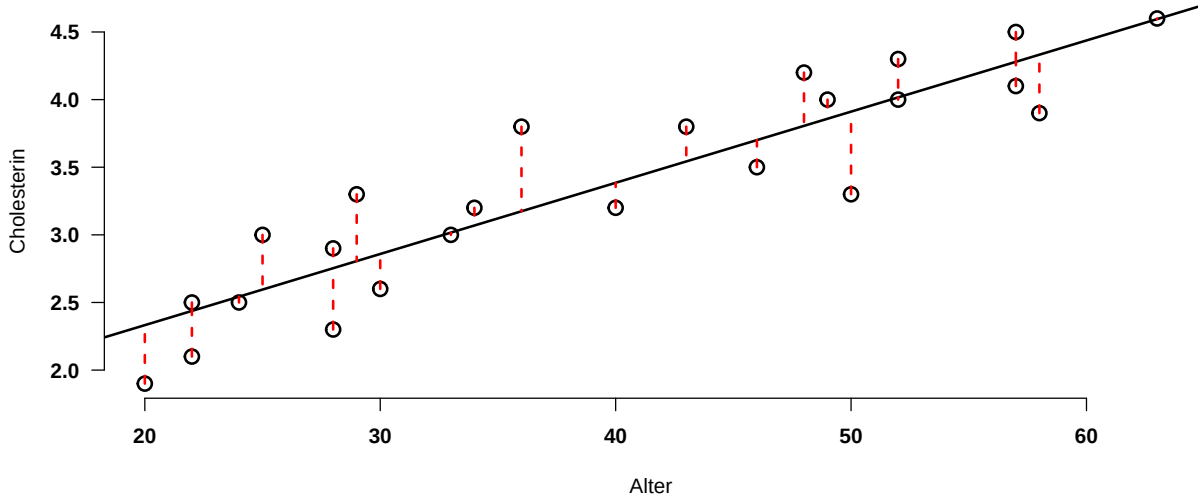


Residuenanalyse:

```
Chol.e <- beta0 + beta1*Alter; e <- Chol - Chol.e

par(lwd=2, font.axis=2, bty="n", ps=11, mfrow=c(1,2))
qqnorm(e, xlab = "Normal-Plot", ylab = "Residuen", las=1, cex=1.5,
                ylim=c(-0.6,+0.6), main="" )
lines(c(-2, +2),c(-0.6, +0.6), lty=2)
plot(Chol.e, e, ylab="Residuen", las=1, cex=1.5,
              xlab="Cholesterin geschätzt", ylim=c(-0.6,+0.6))
abline(h=0, lty=2)
```



```r
Chol.hat <- beta0 + beta1 * Alter

par(lwd=2, font.axis=2, bty="n", ps=12, mfrow=c(1,1))
plot(Alter, Chol, ylab="Cholesterin", las=1, cex=1.5)        # Punktwolke
abline(beta0, beta1, col="black")                            # Regressionsgerade
for (i in 1:n)                                               # Residuen
lines(c(Alter[i],Alter[i]), c(Chol[i],Chol.hat[i]), col="red", lty=2)
```



Varianzkomponenten (elementar):

```r
Chol.hat <- beta0 + beta1 * Alter

ss.regression <- sum((Chol.hat - m.y)^2);    ss.regression
## [1] 11.46477
ms.regression <- ss.regression
ss.residual   <- sum((Chol.hat - Chol)^2);   ss.residual
## [1] 2.454809
ms.residual   <- ss.residual/(n-2)
F.ratio       <- ms.regression/ms.residual;  F.ratio
## [1] 102.7473

ss.regression + ss.residual;                 ss.yy
## [1] 13.91958
## [1] 13.91958

r.cor <- sqrt(ss.regression/ss.yy);           r.cor          # Korrelationskoeffizient
## [1] 0.9075481

cor(Alter, Chol)
## [1] 0.9075481
```

Statistische Eigenschaften der Schätzung (Matrizen):

Funktion ginv() in library(MASS)

```
library(MASS)
y <- Chol;  x <- Alter;   n <- length(x)
r <- Chol - Chol.hat

data <- cbind(x, y, r); as.data.frame(data[1:10,])
```

x	y	r
46	3.5	-0.2006140
20	1.9	-0.4323664
52	4.0	-0.0163634
30	2.6	-0.2586155
57	4.5	0.2205121
25	3.0	0.4045090
28	2.9	0.1466343
36	3.8	0.6256351
22	2.1	-0.3376162
43	3.8	0.2572608

```
X <- cbind(rep(1,n), x)

SSx <- ginv(t(X)%*%X)                                    # SAQ x

beta <- SSx %*% t(X) %*% y; beta                         # Koeffizienten
##             [,1]
## [1,] 1.2798684
## [2,] 0.0526249

SSr <- t(r)%*%r; SSr                                     # SAQ in den Residuen
##           [,1]
## [1,] 2.454809

MSr <- SSr/(n-2); MSr                                    # MSQ in den Residuen
##             [,1]
## [1,] 0.1115822

sqrt(SSr/(n-2))*sqrt(SSx[2,2])                           # Standardfehler von beta1
##               [,1]
## [1,] 0.005191658

sqrt(SSr/(n-2))*sqrt(SSx[1,1])                           # Standardfehler von beta0
##             [,1]
## [1,] 0.2156987

SSy <-  t(beta) %*% t(X) %*% y
MSy <- SSy - n*(mean(y)^2); MSy                          # MSQ in der Regression
##           [,1]
## [1,] 11.46477
```

Lineares Modell in R mit der Funktion lm():

```
lin.model <- lm(Chol ~ Alter)
summary(lin.model)
##
## Call:
## lm(formula = Chol ~ Alter)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.6111 -0.2151 -0.0058  0.2297  0.6256
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1.279868   0.215699   5.934 5.69e-06 ***
## Alter       0.052625   0.005192  10.136 9.43e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.334 on 22 degrees of freedom
## Multiple R-squared:  0.8236, Adjusted R-squared:  0.8156
## F-statistic: 102.7 on 1 and 22 DF,  p-value: 9.428e-10

anova(lin.model)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Alter	1	11.464774	11.4647740	102.7473	0
Residuals	22	2.454809	0.1115822	NA	NA

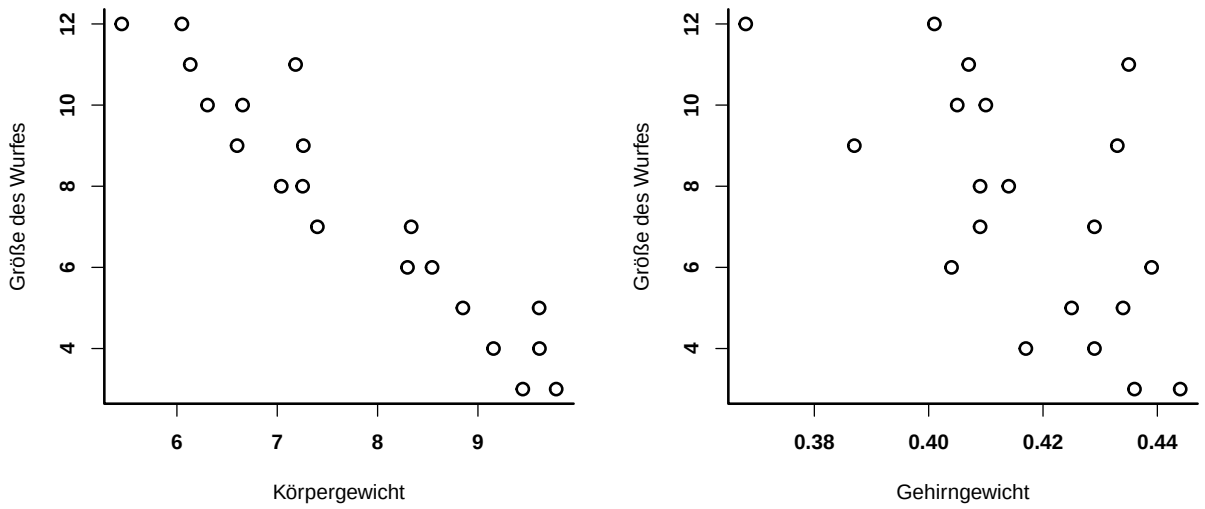
8.2.2 Multiple lineare Regression

Beispiel Körpergewicht, Hirngewicht und Wurfgröße bei Mäusen (data(litters) in library(DAAG)):

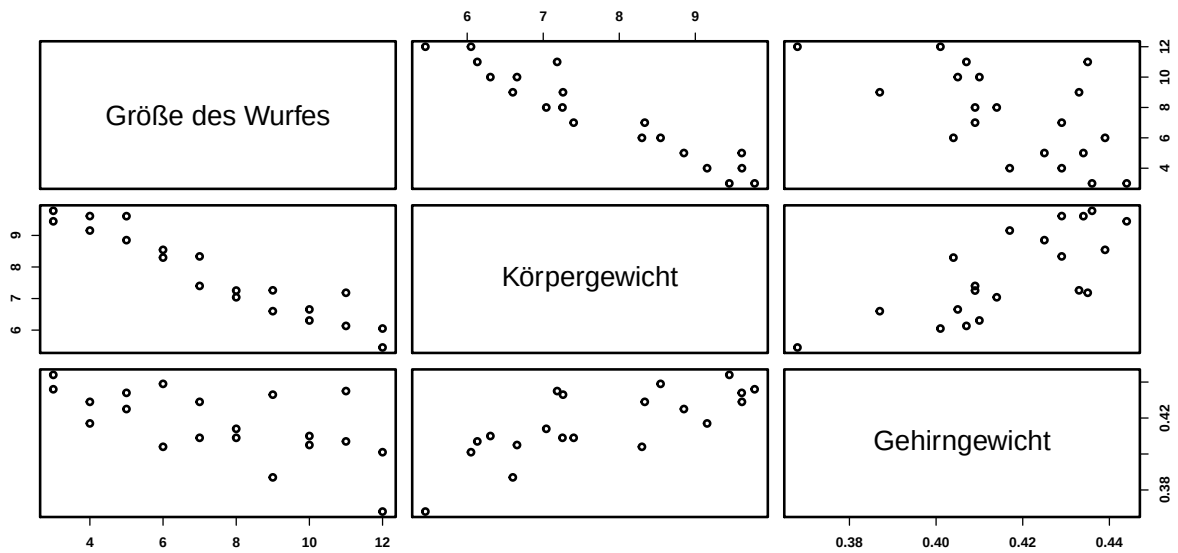
```
library(DAAG)
data(litters)
attach(litters)

litters[1:10,]
```

lsize	bodywt	brainwt
3	9.447	0.444
3	9.780	0.436
4	9.155	0.417
4	9.613	0.429
5	8.850	0.425
5	9.610	0.434
6	8.298	0.404
6	8.543	0.439
7	7.400	0.409
7	8.335	0.429

```
library(DAAG)
data(litters)
attach(litters)

par(mfrow=c(1,2), lwd=2, font.axis=2, bty="l", ps=12)
plot(bodywt, lsize, ylab="Größe des Wurfes",
                xlab="Körpergewicht", cex=1.3)
plot(brainwt, lsize, ylab="Größe des Wurfes",
                xlab="Gehirngewicht", cex=1.3)
```




```
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="o", ps=12)
pairs(litters, labels=c("Größe des Wurfes","Körpergewicht","Gehirngewicht"))
```




```
y <- lsize;  n <- length(y)
X <- matrix(c(rep(1,20),bodywt,brainwt),nrow=20,
            dimnames = list(1:20, c("one", "bodywt", "brainwt"))); p <- 2

t(X) %*% X                                              # Matrixprodukt
##              one     bodywt   brainwt
## one       20.000  154.96000  8.335000
## bodywt   154.960 1235.85592 64.948762
## brainwt    8.335   64.94876  3.480561
```

```
xtxi <- solve(t(X) %*% X); xtxi
##                  one      bodywt     brainwt
## one       38.3034161  0.92161959 -108.924114
## bodywt     0.9216196  0.06404389   -3.402116
## brainwt -108.9241143 -3.40211562  324.615971
```

```
b.h <- xtxi %*% t(X) %*% y; b.h                    #  Parameterschätzung
##               [,1]
## one      12.898778
## bodywt   -2.398031
## brainwt 31.628479
```

```
H   <- X %*% xtxi %*% t(X)                         # Hut-Matrix

y.h <- H %*% y;
e.h <- y - y.h;                                    # Schätzfehler - Residuen
cbind(y, y.h, e.h)[1:10,]
##    y
## 1  3 4.287621 -1.2876207
## 2  3 3.236048 -0.2360484
## 3  4 4.133877 -0.1338770
## 4  4 3.415120  0.5848797
## 5  5 5.118304 -0.1183044
## 6  5 3.580457  1.4195432
## 7  6 5.777820  0.2221804
## 8  6 6.297299 -0.2972987
## 9  7 8.089394 -1.0893942
## 10 7 6.479804  0.5201956
```

```
RSS <- t(e.h) %*% e.h; RSS                         # Summe über die quadr. Abweichungen
##          [,1]
## [1,] 11.48166

s.h <- sqrt(RSS/(n-p-1))                           # Schätzung der Standardabweichung

se.b  <- sqrt(diag(xtxi))*s.h; se.b                # Standardfehler der Schätzung
##        one     bodywt    brainwt
##  5.0862375  0.20797777 14.8068549

R <- 1 - RSS/sum((y-mean(y))^2);R                  # Berechnung des Bestimmtheitsmaßes
##           [,1]
## [1,] 0.9304142
```

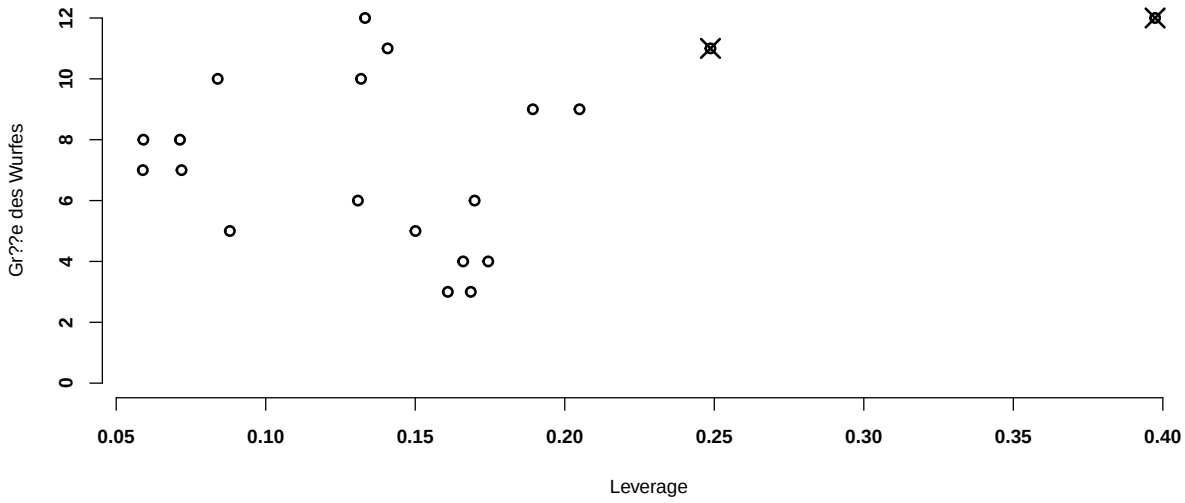
Lineares Modell in R mit der Funktion lm():

```
fit <- lm(lsize ~ bodywt + brainwt, data=litters)
summary(fit)
##
## Call:
## lm(formula = lsize ~ bodywt + brainwt, data = litters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2876 -0.3445 -0.1261  0.5230  1.5679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   12.899      5.086   2.536   0.0213 *
## bodywt        -2.398      0.208 -11.530 1.85e-09 ***
## brainwt       31.628     14.807   2.136   0.0475 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8218 on 17 degrees of freedom
## Multiple R-squared:  0.9304, Adjusted R-squared:  0.9222
## F-statistic: 113.7 on 2 and 17 DF,  p-value: 1.45e-10
```

8.2.4 Analyse der Residuen im linearen Modell

Beispiel Körpergewicht, Hirngewicht und Wurfgröße bei Mäusen (data(litters) in library(DAAG)):

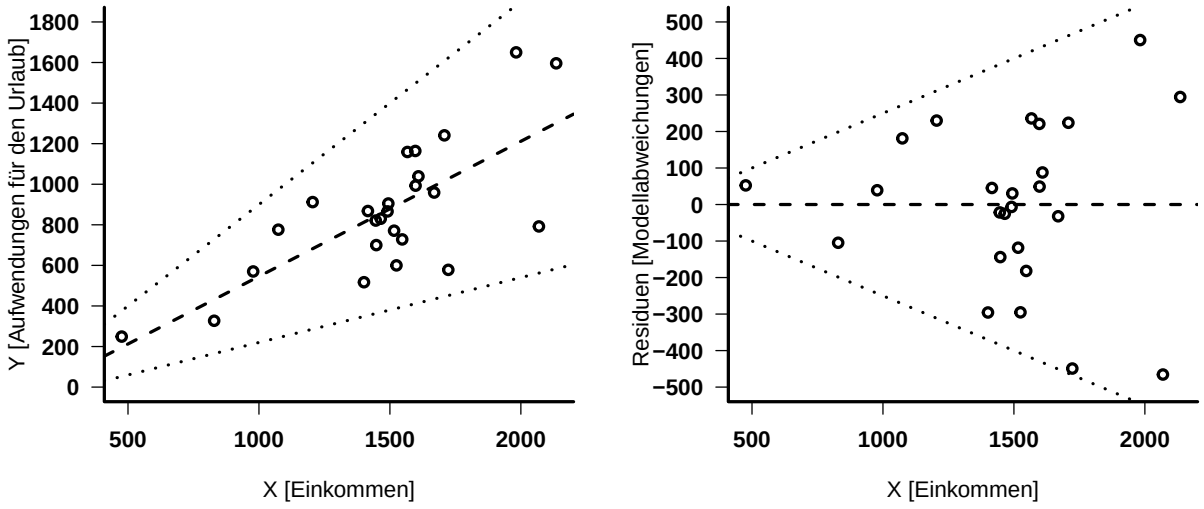
```
library(DAAG)                                          # Influence - Diagnostik
data(litters)

fit <- lm(lsize ~ bodywt + brainwt, data=litters)
res.diag <- influence.measures(fit)
round(cooks.distance(fit), 3)
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## 0.187 0.007 0.002 0.040 0.001 0.207 0.006 0.008 0.039 0.011 0.050 0.006 0.014
##    14    15    16    17    18    19    20
## 0.005 0.048 0.003 0.535 0.000 0.152 0.075
                                                       # Beobachtungen mit 'Einfluss'
infl <- as.numeric(which(apply(res.diag$is.inf,1,any))); infl
## [1] 17 19

par(lwd=2, font.axis=2, bty="n", ps=11, mfrow=c(1,1))
plot(litters$lsize ~ hatvalues(fit), ylim=c(0,12),
       ylab="Gr??e des Wurfes", xlab = "Leverage")
points(hatvalues(fit)[infl], litters$lsize[infl], cex=2, pch=4)
```



8.2.5 Heteroskedastizität im linearen Modell

Beispiel Ausgaben für den Urlaub:

```
x <- c(1516, 1416, 1205, 1547,  829, 1669,  476, 1448, 1982,  978, 1567, 1401,
       1494, 2069, 1609, 1465, 1491, 1598, 1708, 1597, 1446, 1074, 1525, 2135, 1723)
y <- c(771,  868,  912,  728,  327,  959,  249,  700, 1650,  570, 1159,  517,  905,
       792, 1039, 830,  866,  993, 1241, 1164,  821,  776,  600, 1596,  578)

mod <- lm(y~x); summary(mod)                          # lineares Regressionsmodell
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -465.58 -118.27   30.38  181.12  450.36
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -120.4336   196.7850  -0.612    0.547
## x              0.6660     0.1294   5.148 3.23e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 230.4 on 23 degrees of freedom
## Multiple R-squared:  0.5354, Adjusted R-squared:  0.5152
## F-statistic:  26.5 on 1 and 23 DF,  p-value: 3.234e-05

par(mfcol=c(1,2), lwd=2.5, font.axis=2, bty="l", ps=13)
plot(x, y, xlab="X [Einkommen]", ylab="Y [Aufwendungen für den Urlaub]",
     las=1, ylim=c(0,1800), yaxp=c(0,1800,9))
abline(mod, lty=2)
abline(a=-100, b=1.00, lty=3)
abline(a=-100, b=0.32, lty=3)
mod <- lm(y~x); res <- residuals(mod)
plot(x, res, xlab="X [Einkommen]", ylab="Residuen [Modellabweichungen]",
     las=1, ylim=c(-500,500), yaxp=c(-500,+500,10))
abline(h=0, lty=2)
abline(a=-50, b=0.30, lty=3)
abline(a=50, b=-0.30, lty=3)
```



Funktion hccm() in library(car)

```
library(car)
v <- vcov(mod); sqrt(c(v[1,1], v[2,2]))              # Kovarianzmatrix
## [1] 196.7849979   0.1293777
v <- hccm(mod, type="hc0"); sqrt(c(v[1,1], v[2,2]))   # korrigiert nach HC0
## [1] 181.8968175   0.1403881
v <- hccm(mod, type="hc1"); sqrt(c(v[1,1], v[2,2]))   # korrigiert nach HC1
## [1] 189.6405416   0.1463647
v <- hccm(mod, type="hc2"); sqrt(c(v[1,1], v[2,2]))   # korrigiert nach HC2
## [1] 196.5128558   0.1515079
v <- hccm(mod, type="hc3"); sqrt(c(v[1,1], v[2,2]))   # korrigiert nach HC3
## [1] 212.7195847   0.1637564
```

8.2.6 Hypothesentest und Konfidenzintervalle

Beispiel Körpergewicht, Hirngewicht und Wurfgröße bei Mäusen (data(litters) in library(DAAG)):

```
library(DAAG)
data(litters)

fit <- lm(lsize ~ bodywt + brainwt, data=litters)
RSS <- sum(fit$res^2)
SYY <- sum((litters$lsize - mean(litters$lsize))^2)
p   <- 2;   n <- 20
F   <- ((SYY-RSS)/p)/(RSS/(n-p-1)); F
## [1] 113.6513
p   <- 1-pf(F, p, n-p-1); p
## [1] 1.450194e-10
```

```
fit <- lm(lsize ~ bodywt + brainwt, data=litters)
p   <- 2;   n <- 20
x0  <- c(1.0, 8.0, 0.4)                          # einzelne Beobachtung
y0  <- sum(x0*fit$coef)                          # Schätzung der Wurfgröße
X   <- cbind(1, bodywt, brainwt)                 # Aufbau der X-Matrix
xtxi  <- solve(t(X) %*% X)                       # Varianz-Matrix

t     <- qt(0.975, n-p-1)                        # Quantil der t-Verteilung
sigma <- sqrt(sum(fit$res^2)/(n-p-1))            # Schätzung Standardabweichung
W     <- sqrt(1 + x0 %*% xtxi %*% x0)            # Wurzelterm
round(c(y0-t*sigma*W, y0, y0+t*sigma*W), 2)
## [1] 4.49 6.37 8.24
```

8.2.6 Verfahren der Variablenauswahl

Beispiel Körpergewicht, Hirngewicht und Wurfgröße bei Mäusen (data(litters) in library(DAAG)):

```
library(DAAG)
data(litters)

fit <- lm(lsize ~ ., data=litters)
drop1(fit, test="F")
```

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
	NA	NA	11.48166	-5.099626	NA	NA
bodywt	1	89.790832	101.27249	36.441651	132.946294	0.0000000
brainwt	1	3.081674	14.56333	-2.344505	4.562795	0.0475132

```
fit <- lm(lsize ~ 1, data=litters)

add1(fit, ~ bodywt + brainwt, test="F")
```

	Df	Sum of Sq	RSS	AIC	F value	Pr(>F)
	NA	NA	165.00000	44.204264	NA	NA
bodywt	1	150.43667	14.56333	-2.344505	185.93681	0.0000000
brainwt	1	63.72751	101.27249	36.441651	11.32682	0.0034445

```
fit <- lm(lsize ~ ., data=litters)

step(fit)
## Start:  AIC=-5.1
## lsize ~ bodywt + brainwt
##
##           Df Sum of Sq     RSS    AIC
## <none>                  11.482 -5.100
## - brainwt  1     3.082  14.563 -2.345
## - bodywt   1    89.791 101.272 36.442
##
## Call:
## lm(formula = lsize ~ bodywt + brainwt, data = litters)
##
## Coefficients:
## (Intercept)       bodywt      brainwt
##      12.899       -2.398        31.628
```

8.3 Varianzanalyse im linearen Modell

8.3.1 Einfaktorielle Varianzanalyse

Beispiel Wirksamkeit Antibiotika:

```
hemmhof <- read.csv2("hemmhof.CSV")
hemmhof <- as.data.frame(hemmhof)
hemmhof[1:10,]
```

Antibiotikum	Wert
A	13.2
A	14.1
A	7.8
A	11.7
B	15.9
B	16.2
B	19.3
B	18.0
B	17.3
C	6.8

```
summary(hemmhof)
##  Antibiotikum      Wert
##  A:4          Min.   : 6.80
##  B:5          1st Qu.:11.07
##  C:3          Median :13.65
##               Mean   :13.49
##               3rd Qu.:16.48
##               Max.   :19.30
```

8.3.1.1 Erwartungswert-Parametrisierung

```
attach(hemmhof)

Antibiotikum <- as.factor(c(rep("A",4), rep("B",5), rep("C",3)))
Antibiotikum
##  [1] A A A A B B B B B C C C
## Levels: A B C
Wert <- c(13.2,14.1,7.8,11.7,15.9,16.2,19.3,18.0,17.3,6.8,9.2,12.4)
Wert
##  [1] 13.2 14.1  7.8 11.7 15.9 16.2 19.3 18.0 17.3  6.8  9.2 12.4

fit <- lm(Wert ~ Antibiotikum - 1, data=hemmhof)            # Erwartungswerte
summary(fit)
##
## Call:
## lm(formula = Wert ~ Antibiotikum - 1, data = hemmhof)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.900 -1.215 -0.020  1.615  2.933
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## AntibiotikumA   11.700      1.138  10.277 2.85e-06 ***
## AntibiotikumB   17.340      1.018  17.029 3.73e-08 ***
## AntibiotikumC    9.467      1.315   7.201 5.08e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.277 on 9 degrees of freedom
## Multiple R-squared:  0.9803, Adjusted R-squared:  0.9737
## F-statistic: 149.2 on 3 and 9 DF,  p-value: 5.445e-08
```

```
model.matrix(fit)
##    AntibiotikumA AntibiotikumB AntibiotikumC
## 1              1             0             0
## 2              1             0             0
## 3              1             0             0
## 4              1             0             0
## 5              0             1             0
## 6              0             1             0
## 7              0             1             0
## 8              0             1             0
## 9              0             1             0
## 10             0             0             1
## 11             0             0             1
## 12             0             0             1
## attr(,"assign")
## [1] 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Antibiotikum
## [1] "contr.treatment"
```

8.3.1.2 Effekt-Parametrisierung: Dummy-Codierung

```
attach(hemmhof)

fit <- lm(Wert ~ Antibiotikum, contrasts = list(Antibiotikum="contr.treatment"))
summary(fit)
##
## Call:
## lm(formula = Wert ~ Antibiotikum, contrasts = list(Antibiotikum = "contr.treatment"))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.900 -1.215 -0.020  1.615  2.933
##
## Coefficients:
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      11.700      1.138  10.277 2.85e-06 ***
## AntibiotikumB     5.640      1.527   3.693  0.00498 **
## AntibiotikumC    -2.233      1.739  -1.284  0.23113
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.277 on 9 degrees of freedom
## Multiple R-squared:  0.7438, Adjusted R-squared:  0.6869
## F-statistic: 13.07 on 2 and 9 DF,  p-value: 0.002179
```

```
contr.treatment(3, base = 1, contrasts = TRUE)             # Codierung von Kontrasten
##   2 3
## 1 0 0
## 2 1 0
## 3 0 1

summary(aov(Wert ~ Antibiotikum))
##              Df Sum Sq Mean Sq F value  Pr(>F)
## Antibiotikum  2 135.49   67.75   13.07 0.00218 **
## Residuals     9  46.66    5.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model.matrix(fit)
##    (Intercept) AntibiotikumB AntibiotikumC
## 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             1             0
## 10           1             0             1
## 11           1             0             1
## 12           1             0             1
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Antibiotikum
## [1] "contr.treatment"
```

8.3.1.3 Effekt-Parametrisierung: Effekt-Codierung

```
fit <- lm(Wert ~ Antibiotikum, contrasts = list(Antibiotikum="contr.sum"))

summary(fit)
##
## Call:
```

```
## lm(formula = Wert ~ Antibiotikum, contrasts = list(Antibiotikum = "contr.sum"))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.900 -1.215 -0.020  1.615  2.933
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    12.8356     0.6717  19.108 1.36e-08 ***
## Antibiotikum1  -1.1356     0.9398  -1.208 0.257729
## Antibiotikum2   4.5044     0.8927   5.046 0.000694 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.277 on 9 degrees of freedom
## Multiple R-squared:  0.7438, Adjusted R-squared:  0.6869
## F-statistic: 13.07 on 2 and 9 DF,  p-value: 0.002179
```

```
contr.sum(3, contrasts = TRUE)                              # Codierung von Kontrasten
##   [,1] [,2]
## 1    1    0
## 2    0    1
## 3   -1   -1
```

```
model.matrix(fit)
##    (Intercept) Antibiotikum1 Antibiotikum2
## 1            1             1             0
## 2            1             1             0
## 3            1             1             0
## 4            1             1             0
## 5            1             0             1
## 6            1             0             1
## 7            1             0             1
## 8            1             0             1
## 9            1             0             1
## 10           1            -1            -1
## 11           1            -1            -1
## 12           1            -1            -1
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Antibiotikum
## [1] "contr.sum"
```

8.3.1.4 Varianzkomponenten (ANOVA)

Funktion glht() in library(multcomp)

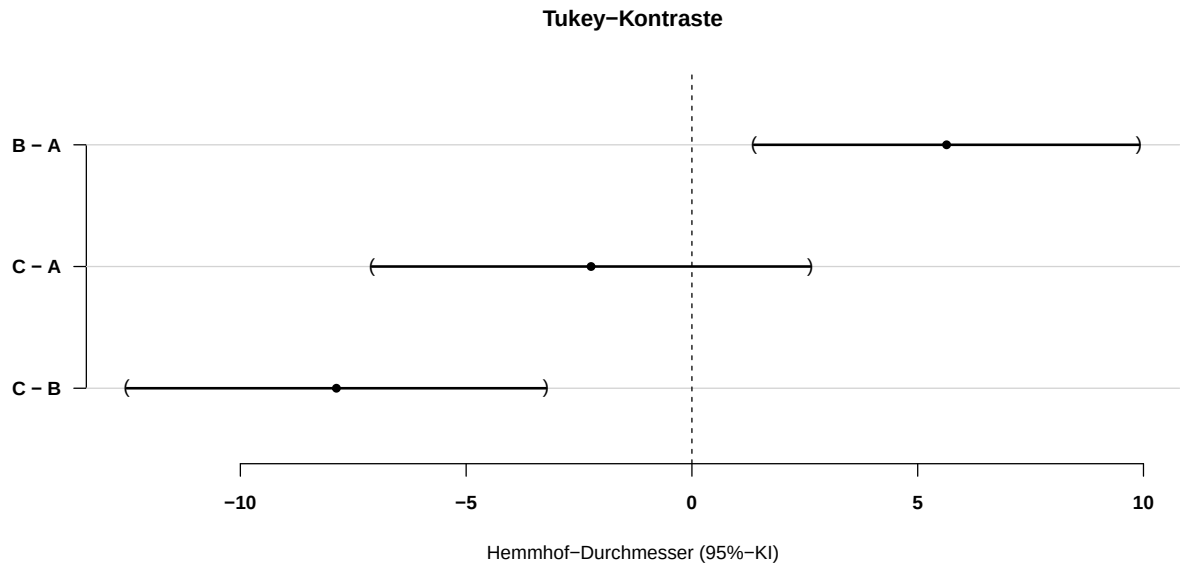
```
library(multcomp)

anova(lm(Wert~Antibiotikum))                                    #  Varianzkomponenten
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Antibiotikum	2	135.49050	67.745250	13.0674	0.0021791
Residuals	9	46.65867	5.184296	NA	NA

Simultane Konfidenzintervalle:

```
d    <- data.frame(Gruppe = factor(Antibiotikum), Wert)
amod <- aov(Wert ~ Antibiotikum, data=d)
summary(glht(amod, linfct = mcp(Antibiotikum = "Tukey")))
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: aov(formula = Wert ~ Antibiotikum, data = d)
##
## Linear Hypotheses:
##             Estimate Std. Error t value Pr(>|t|)
## B - A == 0    5.640      1.527   3.693  0.01225 *
## C - A == 0   -2.233      1.739  -1.284  0.43730
## C - B == 0   -7.873      1.663  -4.735  0.00281 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

```
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="n", ps=11)
plot(glht(amod, linfct = mcp(Antibiotikum = "Tukey")),
     main="Tukey-Kontraste",  xlab="Hemmhof-Durchmesser (95%-KI)")
```



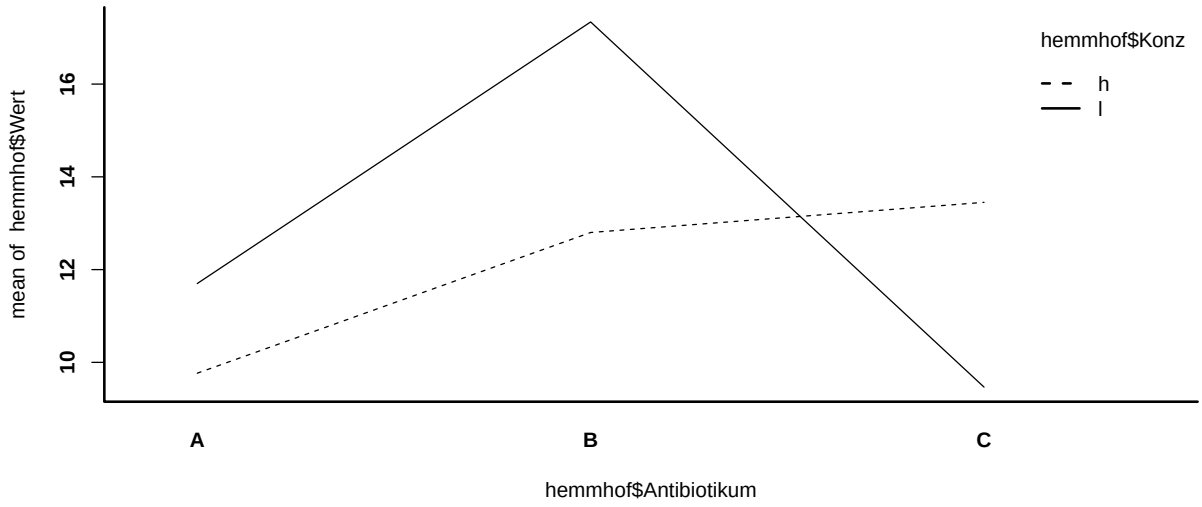
8.3.2 Zweifaktorielle Varianzanalyse

Beispiel Wirksamkeit Antibiotika (2):

```
hemmhof <- read.csv2("hemmhof2.CSV")
hemmhof <- as.data.frame(hemmhof)
hemmhof[1:10,]
```

Antibiotikum	Konz	Wert
A	l	13.2
A	l	14.1
A	l	7.8
A	l	11.7
B	l	15.9
B	l	16.2
B	l	19.3
B	l	18.0
B	l	17.3
C	l	6.8

```
hemmhof <- read.csv2("hemmhof2.CSV")
hemmhof <- as.data.frame(hemmhof)
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=12)
interaction.plot(hemmhof$Antibiotikum, hemmhof$Konz, hemmhof$Wert)
```



Modell ohne Interaktion:

```
fit <- lm(Wert~ Antibiotikum + Konz, data=hemmhof,
          contrasts=list(Antibiotikum="contr.sum", Konz="contr.sum"))
summary(fit)
##
## Call:
## lm(formula = Wert ~ Antibiotikum + Konz, data = hemmhof, contrasts = list(Antibiotikum = "contr.sum"
##     Konz = "contr.sum"))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6077 -2.0758  0.3299  1.9953  5.4558
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    12.6240     0.6344  19.900 3.49e-14 ***
## Antibiotikum1  -1.8356     0.9167  -2.002  0.05973 .
## Antibiotikum2   2.6336     0.8612   3.058  0.00647 **
## Konz1          -0.5817     0.6350  -0.916  0.37109
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.019 on 19 degrees of freedom
## Multiple R-squared:  0.3628, Adjusted R-squared:  0.2622
## F-statistic: 3.607 on 3 and 19 DF,  p-value: 0.03242

anova(fit)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Antibiotikum	2	90.97215	45.486073	4.9904587	0.0181173
Konz	1	7.64944	7.649440	0.8392506	0.3710930

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Residuals	19	173.17754	9.114608	NA	NA

```
model.matrix(fit)
##    (Intercept) Antibiotikum1 Antibiotikum2 Konz1
## 1            1             1             0    -1
## 2            1             1             0    -1
## 3            1             1             0    -1
## 4            1             1             0    -1
## 5            1             0             1    -1
## 6            1             0             1    -1
## 7            1             0             1    -1
## 8            1             0             1    -1
## 9            1             0             1    -1
## 10           1            -1            -1    -1
## 11           1            -1            -1    -1
## 12           1            -1            -1    -1
## 13           1             1             0     1
## 14           1             1             0     1
## 15           1             1             0     1
## 16           1             0             1     1
## 17           1             0             1     1
## 18           1             0             1     1
## 19           1             0             1     1
## 20           1            -1            -1     1
## 21           1            -1            -1     1
## 22           1            -1            -1     1
## 23           1            -1            -1     1
## attr(,"assign")
## [1] 0 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$Antibiotikum
## [1] "contr.sum"
##
## attr(,"contrasts")$Konz
## [1] "contr.sum"
```

Test auf Interaktion:

```
fit1 <- update(fit, . ~ . + Antibiotikum:Konz)
anova(fit, fit1)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
19	173.1775	NA	NA	NA	NA
17	101.4153	2	71.76221	6.01466	0.0105853

Modell mit Interaktion:

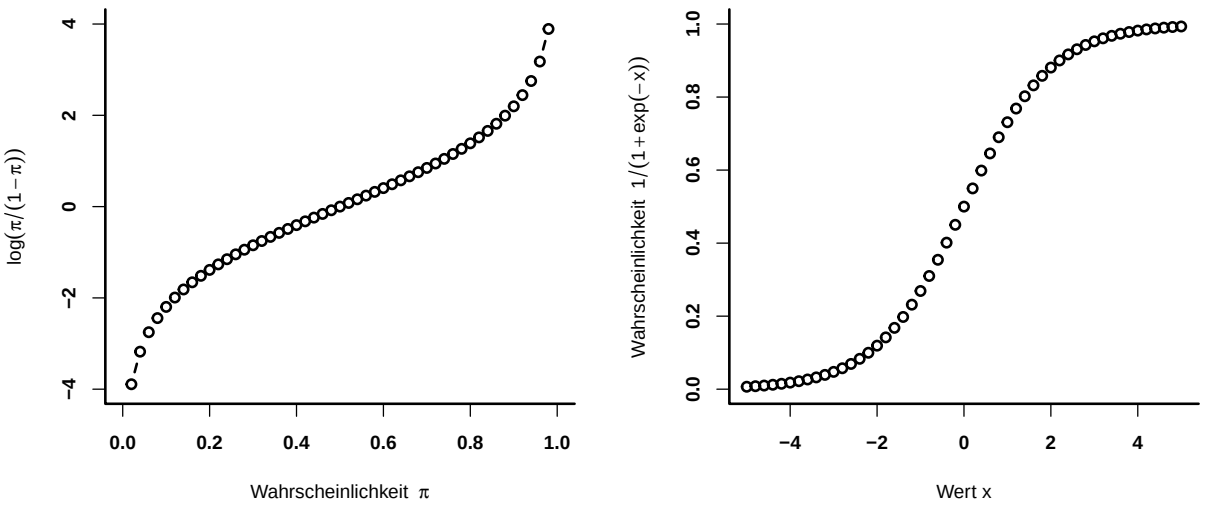
```
fit <- lm(Wert~ Antibiotikum + Konz + Antibiotikum:Konz, data=hemmhof)
anova(fit)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Antibiotikum	2	90.97215	45.486073	7.624717	0.0043289
Konz	1	7.64944	7.649440	1.282257	0.2731996
Antibiotikum:Konz	2	71.76221	35.881105	6.014660	0.0105853
Residuals	17	101.41533	5.965608	NA	NA

8.3 Logistische Regression

Logit-Transformation:

```
p        <- seq(0, 1, 0.02)
logit.p  <- log(p/(1-p))
x        <- seq(-5, +5, 0.2)
ilogit.x <- 1 / (1+exp(-x))

par(mfrow=c(1,2), lwd=2, font.axis=2, bty="l", ps=11)
plot(p, logit.p, type="b", xlim=c(0,1), ylim=c(-4, +4),
     xlab= expression(paste("Wahrscheinlichkeit  ", pi)),
     ylab= expression(log(pi / (1-pi))))
plot(x, ilogit.x, type="b", xlim=c(-5,+5), ylim=c(0,1),
     xlab= expression(paste("Wert x")),
     ylab= expression(paste("Wahrscheinlichkeit  ", 1/(1 + exp(-x)))))
```



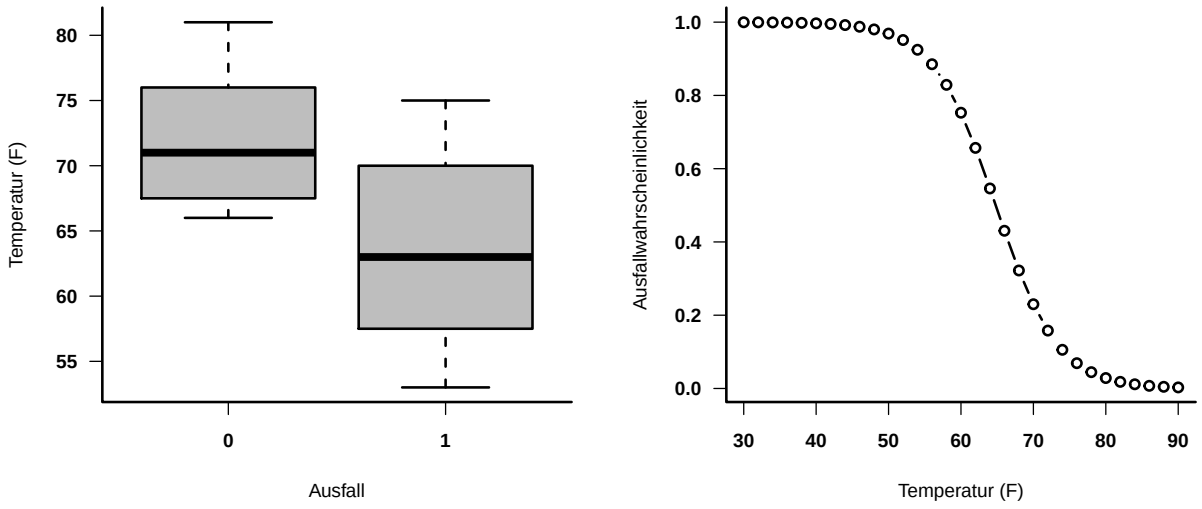
Beispiel Challenger-Unglück:

```
t <- c(66,67,68,70,72,75,76,79,53,58,70,75,67,67,69,70,73,76,78,81,57,63,70)
d <- c( 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1 ,1)

fit <- glm(d ~ t, family=binomial)
summary(fit)
##
## Call:
## glm(formula = d ~ t, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0611  -0.7613  -0.3783   0.4524   2.2175
##
```

```
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  15.0429     7.3786   2.039   0.0415 *
## t            -0.2322     0.1082  -2.145   0.0320 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 28.267  on 22  degrees of freedom
## Residual deviance: 20.315  on 21  degrees of freedom
## AIC: 24.315
##
## Number of Fisher Scoring iterations: 5
```

```
fit$coef
## (Intercept)           t
##  15.0429016  -0.2321627
temp <- seq(30,90,by=2)
lin  <- fit$coef[1] + fit$coef[2]*temp
prob <- exp(lin) / (1 + exp(lin))
```

```
par(mfrow=c(1,2), lwd=2, font.axis=2, bty="l", ps=11)

boxplot(t ~ d, ylab="Temperatur (F)", xlab="Ausfall", las=1, col="grey")
plot(temp, prob, type="b", las=1, xlab="Temperatur (F)",
                  ylab="Ausfallwahrscheinlichkeit")
```



8.4.1 Hypothesentest im linearen Modell

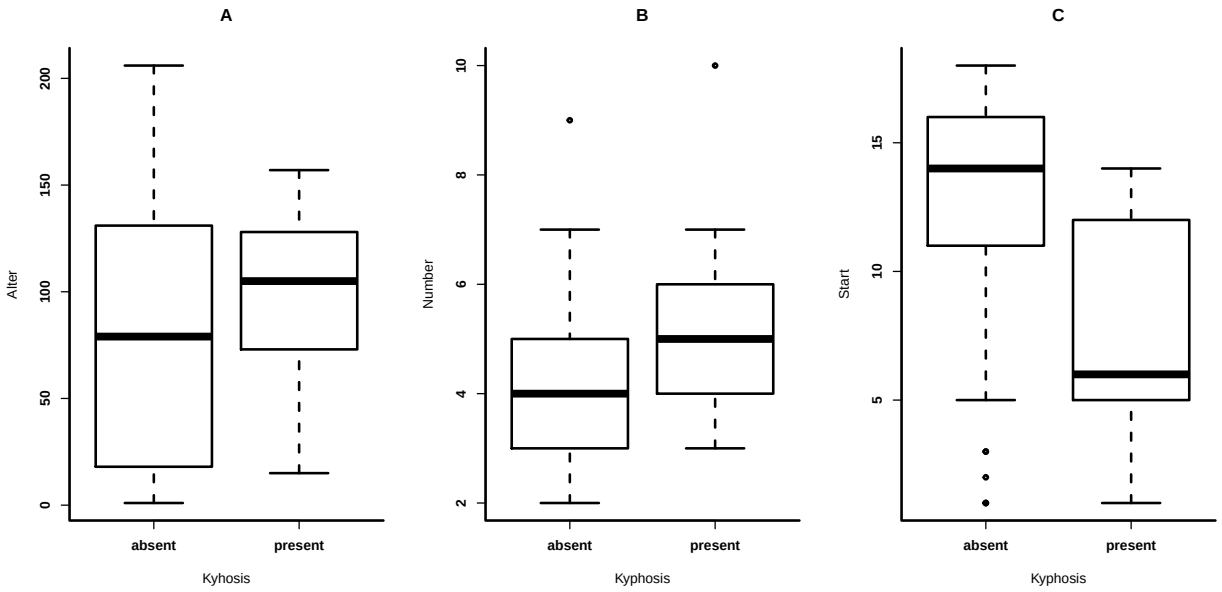
```
anova(fit, test="Chi")
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL	NA	NA	22	28.26715	NA
t	1	7.95196	21	20.31519	0.0048035

8.4.2 Multiple logistische Regression

Beispiel Wirbelsäulenverkrümmung (data(kyphosis) in library(rpart))

```
library(rpart)
attach(kyphosis)

par(mfrow=c(1,3), lwd=2, font.axis=2, bty="l", ps=12)
boxplot(Age~Kyphosis, ylab="Alter", xlab="Kyhosis", main="A")
boxplot(Number~Kyphosis, ylab="Number", xlab="Kyphosis", main="B")
boxplot(Start~Kyphosis, ylab="Start", xlab="Kyphosis", main="C")
```




```
fit <- glm(Kyphosis ~ Age + Number + Start, family="binomial", data=kyphosis)
summary(fit)
##
## Call:
## glm(formula = Kyphosis ~ Age + Number + Start, family = "binomial",
##     data = kyphosis)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.3124  -0.5484  -0.3632  -0.1659   2.1613
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.036934   1.449575  -1.405  0.15996
## Age          0.010930   0.006446   1.696  0.08996 .
## Number       0.410601   0.224861   1.826  0.06785 .
## Start       -0.206510   0.067699  -3.050  0.00229 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 83.234  on 80  degrees of freedom
## Residual deviance: 61.380  on 77  degrees of freedom
## AIC: 69.38
##
## Number of Fisher Scoring iterations: 5
```

```
anova(fit, test="Chi")
```

	Df	Deviance	Resid. Df	Resid. Dev	Pr(>Chi)
NULL	NA	NA	80	83.23447	NA
Age	1	1.301985	79	81.93249	0.2538510
Number	1	10.305931	78	71.62656	0.0013260
Start	1	10.246632	77	61.37993	0.0013693

```
fit1 <- update(fit, . ~ .- Age)
anova(fit, fit1, test="Chi")
```

Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
77	61.37993	NA	NA	NA
78	64.53647	-1	-3.156541	0.0756233

8.4.3 Interpretation der Modellparameter

Modellrechnungen - Vorhersage:

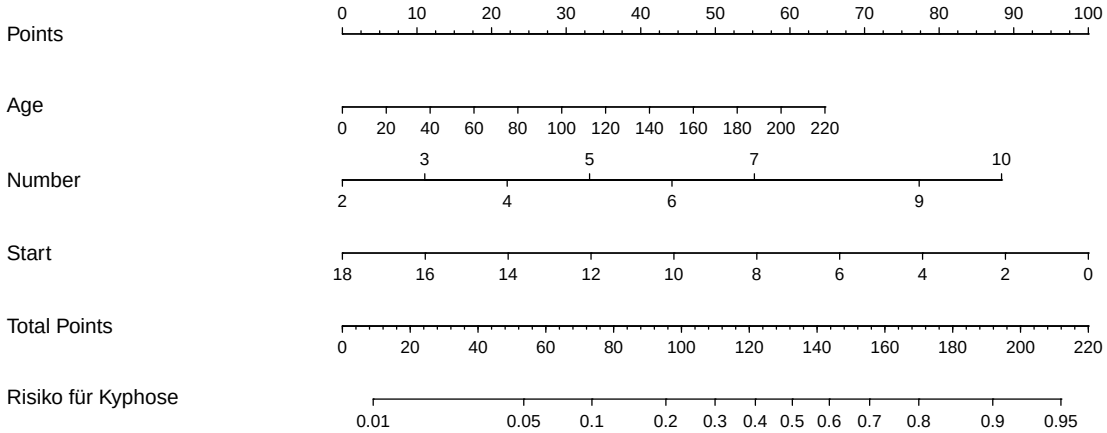
```
fit <- glm(Kyphosis ~ Age + Number + Start, family="binomial", data=kyphosis)

new.d <- data.frame(Age=c(12,24,60), Number=c(2, 4, 6), Start=c(15, 10, 5))
new.p <- round(predict(fit, new.d, type = "response"), 4)

cbind(new.d, new.p)
```

Age	Number	Start	new.p
12	2	15	0.0150
24	4	10	0.1000
60	6	5	0.5125

Funktion nomogram() in library(rms):

Beispiel Wirbelsäulenverkrümmung (data(kyphosis) in library(rpart))

```
library(rpart); library(rms)
data(kyphosis); df <- kyphosis

ddist <- datadist(df); options(datadist='ddist')
fit <- lrm(Kyphosis ~ Age + Number + Start, data=df, x=TRUE, y=TRUE)

nom <- nomogram(fit, fun=function(x)1/(1+exp(-x)),
                fun.at=c(.01,.05,seq(.1,.9,by=.1),.95,.99),
                lp=FALSE, funlabel="Risiko für Kyphose")

plot(nom, xfrac=.45)
```



```
print(nom)
## Points per unit of linear predictor: NaN
## Linear predictor units per point   : NaN
##
##
##  Age Points
##    0  0
##   20  6
##   40 12
##   60 18
##   80 24
##  100 29
##  120 35
##  140 41
##  160 47
##  180 53
##  200 59
##  220 65
##
##
##  Number Points
##   2      0
##   3     11
##   4     22
##   5     33
##   6     44
##   7     55
##   9     77
##  10     88
##
##
##  Start Points
##   0    100
##   2     89
##   4     78
##   6     67
##   8     56
##  10     44
##  12     33
##  14     22
##  16     11
##  18      0
##
##
##  Total Points Risiko für Kyphose
##             9                0.01
##            53                0.05
##            74                0.10
##            95                0.20
##           110                0.30
##           122                0.40
##           133                0.50
##           144                0.60
```

```
##           155              0.70
##           170              0.80
##           192              0.90
##           212              0.95
```

8.4.4 Variablenauswahl (Schrittweise)

Funktion stepAIC() in library(MASS):

Beispiel Wirbelsäulenverkrümmung (data(kyphosis) in library(rpart))

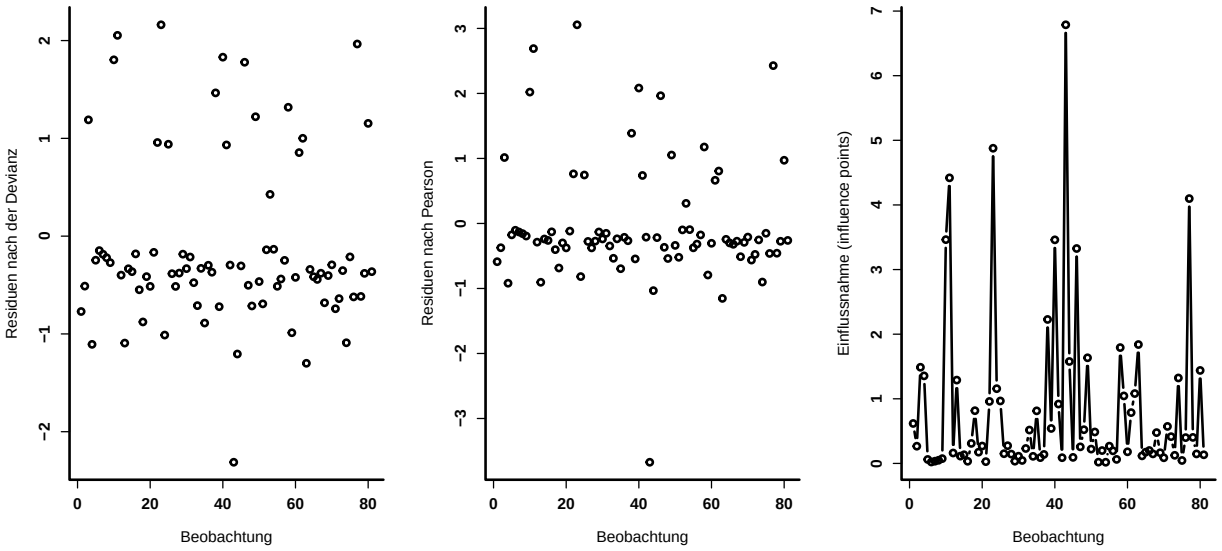
```
library(rpart); data(kyphosis)
library(MASS)

model <- glm(Kyphosis ~ 1,  family = binomial, data = kyphosis); model
##
## Call:  glm(formula = Kyphosis ~ 1, family = binomial, data = kyphosis)
##
## Coefficients:
## (Intercept)
##      -1.326
##
## Degrees of Freedom: 80 Total (i.e. Null);  80 Residual
## Null Deviance:       83.23
## Residual Deviance: 83.23     AIC: 85.23

model.step <- stepAIC(model, Kyphosis ~ Age + Number + Start,
                                    trace = FALSE, direction="both")
model.step$anova
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
	NA	NA	80	83.23447	85.23447
+ Start	1	15.162295	79	68.07218	72.07218
+ Number	1	3.535712	78	64.53647	70.53647
+ Age	1	3.156541	77	61.37993	69.37993

8.4.5 Residuenanalyse

Beispiel Wirbelsäulenverkrümmung (data(kyphosis) in library(rpart))

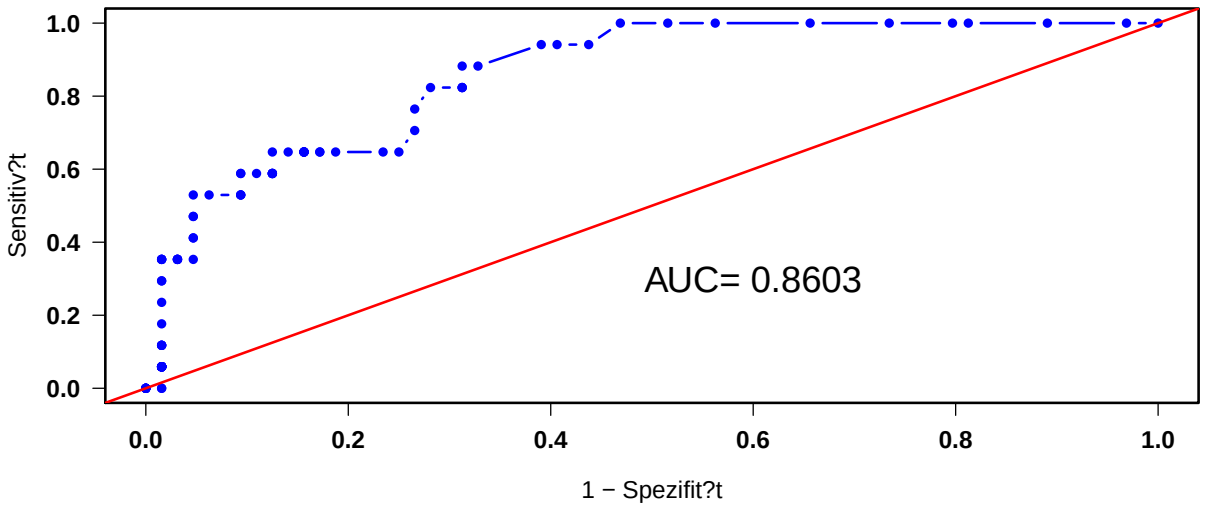
```
library(rpart); data(kyphosis)

fit <- glm(Kyphosis ~ Age + Number + Start, data=kyphosis, family="binomial")
deviance.resid <- residuals(fit)
pearson.resid  <- residuals(fit, type="pearson")
hats <- influence(fit)$hat
idev <- deviance.resid^2 + pearson.resid^2 * hats/(1-hats)

par(mfrow=c(1,3), lwd=2, font.axis=2, bty="l", ps=14)
plot(deviance.resid, xlab="Beobachtung", ylab="Residuen nach der Devianz")
plot(pearson.resid, xlab="Beobachtung", ylab="Residuen nach Pearson")
plot(idev, xlab="Beobachtung",
           ylab="Einflussnahme (influence points)", type="b")
```



8.4.7 Güte der Klassifikation (ROC/AUC)

Beispiel Wirbelsäulenverkrümmung (data(kyphosis) in library(rpart))

```
library(rpart); data(kyphosis)

fit <- glm(Kyphosis ~ Age + Number + Start, family="binomial", data=kyphosis)

x   <- ifelse(kyphosis$Kyphosis=="present", 1, 0)
p.h <- fit$fitted.values

t   <- seq(0, 1, length=100)
rp  <- sapply(t, function(tcut)                          # richtig positiv rp
  { sum(p.h >= tcut & x == 1) / sum(x == 1) } )
t   <- seq(0, 1, length=100)
fp  <- sapply(t, function(tcut)                          # falsch positiv fp
  { sum(p.h >= tcut & x == 0) / sum(x == 0) } )

AUC <- 0
for (i in 1:(length(fp)-1)) {
  step <- abs(fp[i+1] - fp[i]); AUC <- AUC + 0.5*step*(rp[i+1]+rp[i]) }
AUC
## [1] 0.8602941

par(mfrow=c(1,1), lwd=2, font.axis=2, bty="o", ps=14)
plot(fp, rp, type="b", las=1, pch=16, col="blue",
     xlab="1 - Spezifit?t", ylab="Sensitiv?t")
abline(a=0, b=1, col="red")
text(0.6, 0.3, paste("AUC=",round(AUC, 4)), cex=1.5)
```



Funktion performance() in library(ROCR):

```r
library(ROCR)
pred <- prediction(p.h, x)
as.numeric(performance(pred, measure="auc")@y.values)
## [1] 0.859375
```

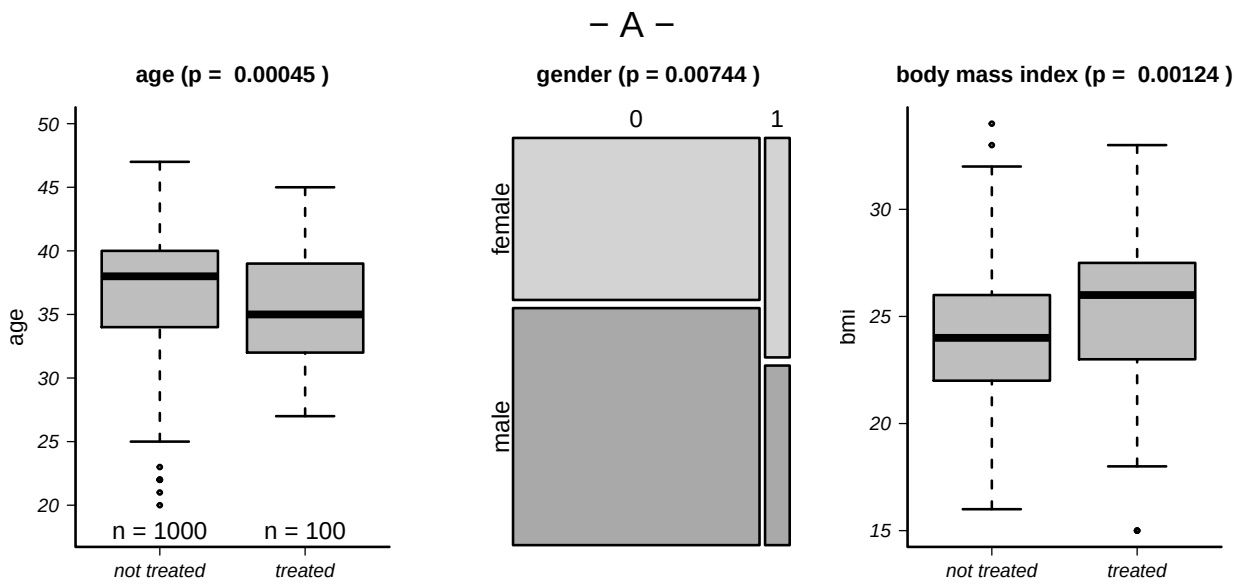
8.4.8 Propensity-Score Matching

```
psd <- read.csv2("psdata.csv")
as.data.frame(psd[1:10,])
```

ident	age	sex	bmi	treat
1	40	female	27	1
2	37	male	19	1
3	30	female	26	1
4	34	male	27	1
5	32	male	15	1
6	31	male	25	1
7	34	female	27	1
8	32	female	15	1
9	33	female	23	1
10	41	male	22	1

```
by(psd$age, psd$treat, summary)
## psd$treat: 0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   34.00   38.00   37.07   40.00   47.00
## ------------------------------------------------------------
## psd$treat: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.00   32.00   35.00   35.61   39.00   45.00
t1 <- wilcox.test(age ~ treat, data=psd); t1
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  age by treat
## W = 60596, p-value = 0.0004524
## alternative hypothesis: true location shift is not equal to 0
p1 <- round(t1$p.value, 5)

tab <- table(psd$sex, psd$treat); tab
##
##            0   1
##   female 406  55
##   male   594  45
t2 <- chisq.test(tab); t2
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab
## X-squared = 7.1629, df = 1, p-value = 0.007443
p2 <- round(t2$p.value, 5)

by(psd$bmi, psd$treat, summary)
## psd$treat: 0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
##   16.00   22.00   24.00   24.02   26.00   34.00
## ------------------------------------------------------------
## psd$treat: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   23.00   26.00   25.06   27.25   33.00
t3 <- wilcox.test(bmi ~ treat, data=psd); t3
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  bmi by treat
## W = 40267, p-value = 0.001244
## alternative hypothesis: true location shift is not equal to 0
p3 <- round(t3$p.value, 5)

par(mfrow=c(1,3), lwd=2, bty="l", font=1, font.axis=3,
    font.lab=1, ps=16, cex.lab=1.2, oma = c(0, 0, 2, 0))
boxplot(age ~ treat, data=psd, col="grey", las=1,
        names = c("not treated","treated"), xlab=" ",
        ylim=c(18,50), main=paste("age (p = ", p1, ")"))
text(x= 1, y= 18, labels= "n = 1000", cex=1.3)
text(x= 2, y= 18, labels= "n = 100", cex=1.3)
mosaicplot(t(tab), col=c("lightgrey","darkgrey"),
           cex.axis=1.3, main=paste("gender (p =",p2,")"))
boxplot(bmi ~ treat, data=psd, col="grey", las=1,
        names = c("not treated","treated"), xlab=" ",
        main=paste("body mass index (p = ", p3, ")"))
mtext("- A -", side=3, outer = TRUE, cex = 1.3)
```



Funktion Match() in library(Matching)

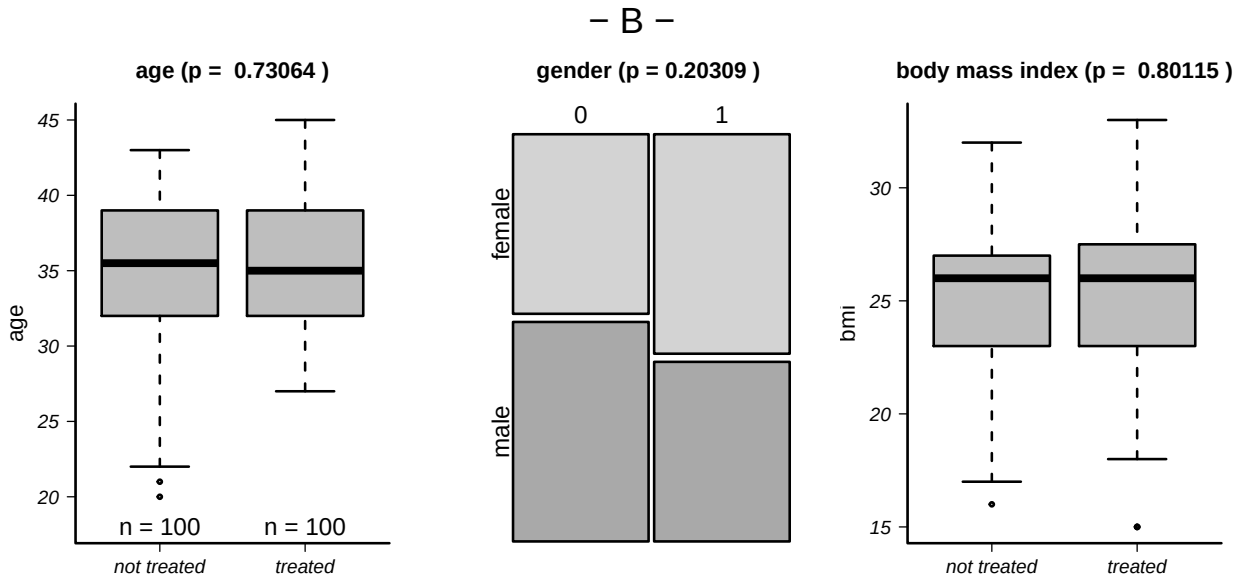
```
library(Matching)
mod  <- glm(treat ~ age + sex + bmi,
            family=binomial, data=psd); summary(mod)
##
## Call:
## glm(formula = treat ~ age + sex + bmi, family = binomial, data = psd)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9153  -0.4764  -0.3941  -0.3223   2.7214
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.10683    1.22399  -1.721  0.08520 .
## age         -0.06378    0.02230  -2.860  0.00424 **
## sexmale     -0.56122    0.21307  -2.634  0.00844 **
## bmi          0.09850    0.03451   2.855  0.00431 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 670.20  on 1099  degrees of freedom
## Residual deviance: 645.05  on 1096  degrees of freedom
## AIC: 653.05
##
## Number of Fisher Scoring iterations: 5

psd$yes <- predict(mod, type = "response")                    # P(treated)
psd$no  <- 1 - psd$yes                                        # P(not treated)

listMatch <- Match(Tr = (psd$treat == 1),
                   X = log(psd$yes/psd$no),                   # matching scale
                   M = 1,                                     # 1:1 matching
                   caliper  = 0.5,                            # 0.5 * SD(X)
                   replace  = FALSE,
                   ties     = TRUE,
                   version  = "fast")
summary(listMatch)
##
## Estimate...  0
## SE.........  0
## T-stat.....  NaN
## p.val......  NA
##
## Original number of observations..............  1100
## Original number of treated obs...............  100
## Matched number of observations...............  100
## Matched number of observations  (unweighted).  100
##
## Caliper (SDs)........................................   0.5
## Number of obs dropped by 'exact' or 'caliper'  0
```

```
treat  <- psd[listMatch$index.treated,]
cntr   <- psd[listMatch$index.control,]
result <- rbind(treat, cntr)
                                                          # Balance
MatchBalance(treat ~ age + sex + bmi, data=psd,
             match.out=listMatch, nboots=500)
##
## ***** (V1) age *****
##                        Before Matching        After Matching
## mean treatment........      35.61              35.61
## mean control.........     37.074              35.08
## std mean diff.........     -34.09             12.342
##
## mean raw eQQ diff.....       1.66               0.61
## med  raw eQQ diff.....          2                  0
## max  raw eQQ diff.....          7                  7
##
## mean eCDF diff........   0.060111           0.022174
## med  eCDF diff........      0.028              0.01
## max  eCDF diff........      0.202              0.07
##
## var ratio (Tr/Co).....    0.90462             0.719
## T-test p-value........  0.0015661            0.16109
## KS Bootstrap p-value.. < 2.22e-16              0.838
## KS Naive p-value......  0.0011996           0.96707
## KS Statistic..........      0.202              0.07
##
##
## ***** (V2) sexmale *****
##                        Before Matching        After Matching
## mean treatment........       0.45               0.45
## mean control.........      0.594              0.55
## std mean diff.........      -28.8               -20
##
## mean raw eQQ diff.....       0.14                0.1
## med  raw eQQ diff.....          0                  0
## max  raw eQQ diff.....          1                  1
##
## mean eCDF diff........      0.072              0.05
## med  eCDF diff........      0.072              0.05
## max  eCDF diff........      0.144               0.1
##
## var ratio (Tr/Co).....     1.0356                 1
## T-test p-value........  0.0068855           0.039629
##
##
## ***** (V3) bmi *****
##                        Before Matching        After Matching
## mean treatment........      25.06              25.06
## mean control.........     24.024              25.24
## std mean diff.........     26.962            -4.6846
##
## mean raw eQQ diff.....       1.24               0.54
```

```
## med  raw eQQ diff.....          1                  0
## max  raw eQQ diff.....          3                  4
##
## mean eCDF diff........     0.0636           0.028421
## med  eCDF diff........     0.0415               0.03
## max  eCDF diff........      0.213               0.06
##
## var ratio (Tr/Co).....     1.6259             1.4411
## T-test p-value........   0.010102            0.55354
## KS Bootstrap p-value.. < 2.22e-16              0.892
## KS Naive p-value...... 0.00052309            0.99376
## KS Statistic..........      0.213               0.06
##
##
## Before Matching Minimum p.value: < 2.22e-16
## Variable Name(s): age bmi  Number(s): 1 3
##
## After Matching Minimum p.value: 0.039629
## Variable Name(s): sexmale  Number(s): 2

# write.csv2(result, "psmatched.csv")                          # Save matched dataset
```

```
by(result$age, result$treat, summary)
## result$treat: 0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   20.00   32.00   35.50   35.08   39.00   43.00
## ------------------------------------------------------------
## result$treat: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   27.00   32.00   35.00   35.61   39.00   45.00
t1 <- wilcox.test(age ~ treat, data=result); t1
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  age by treat
## W = 4859, p-value = 0.7306
## alternative hypothesis: true location shift is not equal to 0
p1 <- round(t1$p.value, 5)

tab <- table(result$sex, result$treat); tab
##
##           0  1
##   female 45 55
##   male   55 45
t2 <- chisq.test(tab); t2
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab
## X-squared = 1.62, df = 1, p-value = 0.2031
p2 <- round(t2$p.value, 5)

by(result$bmi, result$treat, summary)
```

```
## result$treat: 0
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   23.00   26.00   25.24   27.00   32.00
## ------------------------------------------------------------
## result$treat: 1
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   15.00   23.00   26.00   25.06   27.25   33.00
t3 <- wilcox.test(bmi ~ treat, data=result); t3
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  bmi by treat
## W = 5103, p-value = 0.8011
## alternative hypothesis: true location shift is not equal to 0
p3 <- round(t3$p.value, 5)

par(mfrow=c(1,3), lwd=2, bty="l", font=1, font.axis=3,
    font.lab=1, ps=16, cex.lab=1.2, oma = c(0, 0, 2, 0))
boxplot(age ~ treat, data=result, col="grey", las=1,
        names = c("not treated","treated"), xlab=" ",
        ylim=c(18,45), main=paste("age (p = ", p1, ")"))
text(x= 1, y= 18, labels= "n = 100", cex=1.3)
text(x= 2, y= 18, labels= "n = 100", cex=1.3)
mosaicplot(t(tab), col=c("lightgrey","darkgrey"),
           cex.axis=1.3, main=paste("gender (p =",p2,")"))
boxplot(bmi ~ treat, data=result, col="grey", las=1,
        names = c("not treated","treated"), xlab=" ",
        main=paste("body mass index (p = ", p3, ")"))
mtext("- B -", side=3, outer = TRUE, cex = 1.3)
```



8.5 Poisson-Regression und loglineare Modelle

8.5.1 Poisson-Regression

Beispiel Paarungen afrikanischer Elefanten:

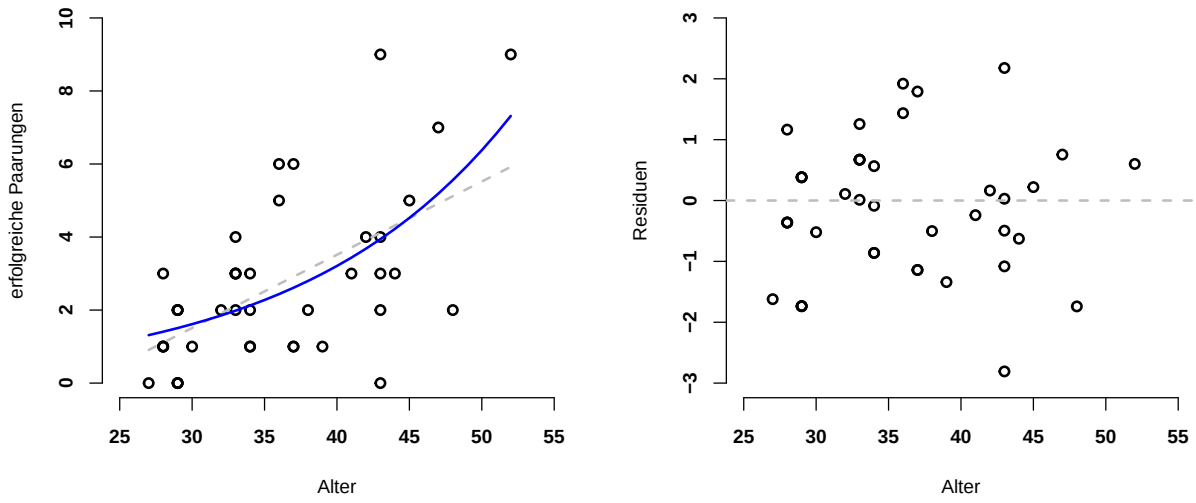
```
alter <- c(27,  28,  28,  28,  28,  29,  29,  29,  29,  29,  29,
           30,  32,  33,  33,  33,  33,  33,  34,  34,  34,  34,
           36,  36,  37,  37,  37,  38,  39,  41,  42,  43,  43,
           43,  43,  43,  44,  45,  47,  48,  52)
paarungen <- c(0,   1,   1,   1,   3,   0,   0,   0,   2,   2,   2,
               1,   2,   4,   3,   3,   3,   2,   1,   1,   2,   3,
               5,   6,   1,   1,   6,   2,   1,   3,   4,   0,   2,
               3,   4,   9,   3,   5,   7,   2,   9)
elephants <- as.data.frame(cbind(alter, paarungen))
elephants[1:5,]
```

alter	paarungen
27	0
28	1
28	1
28	1
28	3

```
                                                    # lineares Modell
linear.mod <- lm(paarungen ~ alter, data=elephants)
summary(linear.mod)
##
## Call:
## lm(formula = paarungen ~ alter, data = elephants)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1158 -1.3087 -0.1082  0.8892  4.8842
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.50589    1.61899  -2.783  0.00826 **
## alter        0.20050    0.04443   4.513 5.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.849 on 39 degrees of freedom
## Multiple R-squared:  0.343,  Adjusted R-squared:  0.3262
## F-statistic: 20.36 on 1 and 39 DF,  p-value: 5.749e-05
```

```
                                                    # Poisson-Regression
poiss.mod <- glm(paarungen ~ alter, family=poisson, data=elephants)
summary(poiss.mod)
##
```

```
## Call:
## glm(formula = paarungen ~ alter, family = poisson, data = elephants)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.80798  -0.86137  -0.08629   0.60087   2.17777
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.58201    0.54462  -2.905  0.00368 **
## alter        0.06869    0.01375   4.997 5.81e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 75.372  on 40  degrees of freedom
## Residual deviance: 51.012  on 39  degrees of freedom
## AIC: 156.46
##
## Number of Fisher Scoring iterations: 5

residuals(poiss.mod, type="pearson")                    # Residuen
##          1          2          3          4          5          6
## -1.14608174 -0.34305096 -0.34305096 -0.34305096  1.34310580 -1.22757632
##          7          8          9         10         11         12
## -1.22757632 -1.22757632  0.40165029  0.40165029  0.40165029 -0.48336228
##         13         14         15         16         17         18
##  0.10890047  1.43181668  0.72177200  0.72177200  0.72177200  0.01172731
##         19         20         21         22         23         24
## -0.77150327 -0.77150327 -0.08543201  0.60063925  1.64140826  2.28193361
##         25         26         27         28         29         30
## -0.99687309 -0.99687309  2.09762252 -0.47622609 -1.15285259 -0.23536712
##         31         32         33         34         35         36
##  0.16645667 -1.98554099 -0.97825885 -0.47461777  0.02902330  2.54722868
##         37         38         39         40         41
## -0.59501233  0.22430326  0.79498239 -1.50921328  0.62273201
new  <- data.frame(alter = seq(27, 52, 1))
linp <- predict(linear.mod, new)
poip <- predict(poiss.mod, new, type = "response")

par(lwd=2, font.axis=2, bty="n", ps=11, mfrow=c(1,2))
plot(alter, paarungen, xlab="Alter", ylab="erfolgreiche Paarungen",
     xlim = c(25, 55), ylim = c(0, 10))
lines(new$alter, linp, col="grey", lty=2)
lines(new$alter, poip, col="blue")

plot(alter, residuals(poiss.mod), xlab="Alter", ylab="Residuen",
     xlim = c(25, 55), ylim = c(-3, +3))
abline(h=0, col="grey", lty=2)
```



8.5.1.1 Dispersionsindex

```
DP <- sum(residuals(poiss.mod, type="pearson")^2)/poiss.mod$df.res; DP
## [1] 1.157334

summary(poiss.mod, dispersion=DP)
##
## Call:
## glm(formula = paarungen ~ alter, family = poisson, data = elephants)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.80798  -0.86137  -0.08629   0.60087   2.17777
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.58201    0.58590  -2.700  0.00693 **
## alter        0.06869    0.01479   4.645  3.4e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1.157334)
##
##     Null deviance: 75.372  on 40  degrees of freedom
## Residual deviance: 51.012  on 39  degrees of freedom
## AIC: 156.46
##
## Number of Fisher Scoring iterations: 5
```

8.5.2 Relative Risiken aus Raten

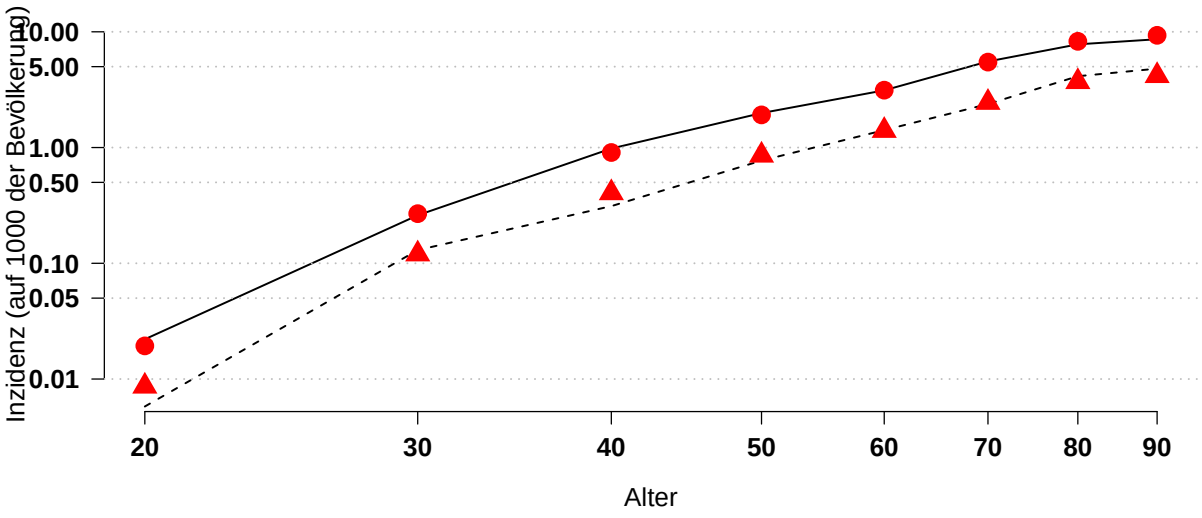
Beispiel Hautkrebsrisiko in zwei US-Städten:

```
rates <- read.csv2("skincancer.csv")
as.data.frame(rates[1:5,])
```

age	region	city	cases	popul	incid
20	1	Minneapolis	1	172675	0.00579
30	1	Minneapolis	16	123065	0.13001
40	1	Minneapolis	30	96216	0.31180
50	1	Minneapolis	71	92051	0.77131
60	1	Minneapolis	102	72159	1.41355

```
rates$city <- relevel(rates$city, ref="Minneapolis")

mod <- glm(cases ~ as.factor(rates$age) + city + offset(log(popul)) - 1,
           family="poisson", data=rates)
summary(mod)
##
## Call:
## glm(formula = cases ~ as.factor(rates$age) + city + offset(log(popul)) -
##     1, family = "poisson", data = rates)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5043  -0.4816   0.0169   0.3697   1.2504
##
## Coefficients:
##                           Estimate Std. Error z value Pr(>|z|)
## as.factor(rates$age)20 -11.65787    0.44871  -25.98   <2e-16 ***
## as.factor(rates$age)30  -9.02768    0.14127  -63.91   <2e-16 ***
## as.factor(rates$age)40  -7.81052    0.09053  -86.27   <2e-16 ***
## as.factor(rates$age)50  -7.06269    0.06984 -101.13   <2e-16 ***
## as.factor(rates$age)60  -6.57059    0.06467 -101.59   <2e-16 ***
## as.factor(rates$age)70  -6.01246    0.05993 -100.33   <2e-16 ***
## as.factor(rates$age)80  -5.59933    0.06327  -88.50   <2e-16 ***
## as.factor(rates$age)90  -5.47969    0.10365  -52.87   <2e-16 ***
## cityDallas               0.80428    0.05221   15.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2.750e+06  on 16  degrees of freedom
## Residual deviance: 8.195e+00  on  7  degrees of freedom
## AIC: 120.44
##
## Number of Fisher Scoring iterations: 4
```

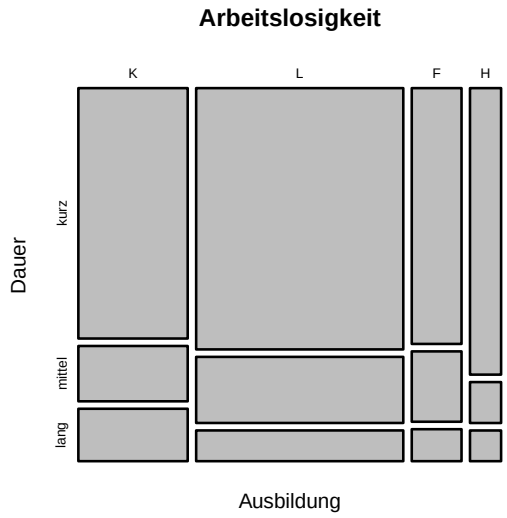
```r
alpha <- c(mod$coefficients[1:8])                    # Zuordnung der Koeffizienten
delta <- mod$coefficients[9]; rr <- exp(delta)
estim1 <- exp(alpha)*1000                            # Inzidenzen für Mineapolis
estim2 <- exp(alpha + delta)*1000                    # Inzidenzen für Dallas

city1  <- rates[rates$city=="Minneapolis",]
city2  <- rates[rates$city=="Dallas",]
resid1 <- city1$incid - estim1; resid2 <- city2$incid - estim2

par(lwd=1.5, font.axis=2, bty="n", ps=15, mfrow=c(1,1))
plot(city2$age, city2$incid, type="l", las=1, lty=1, pch=1, cex=1.2,
     xlab="Alter", ylab="Inzidenz (auf 1000 der Bevölkerung)",
     ylim=c(0.007, 10), log="xy")
abline(h=c(0.01,0.05,0.10,0.5,1.0,5.0,10.0), lty=3, col="grey")
points(city2$age, estim2, pch=16, cex=2, col="red")
lines(city1$age, city1$incid, type="l", lty=2, pch=2, cex=1.2)
points(city1$age, estim1, pch=17, cex=2, col="red")
```

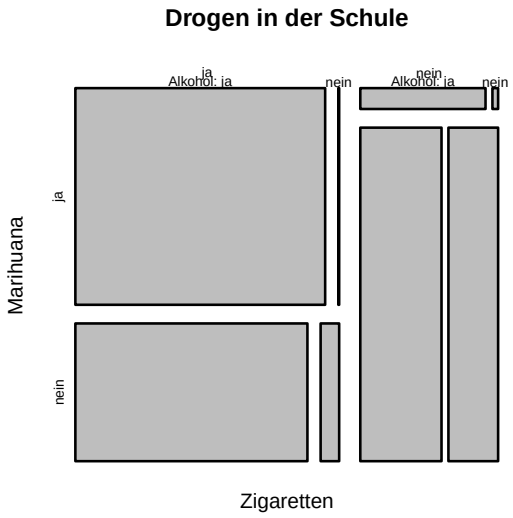


8.5.3 Analyse von Kontingenztafeln

Beispiele Arbeitslosigkeit und Drogenmissbrauch:

```
par(mfrow=c(1,2), lwd=2, font.axis=3, bty="l", ps=12)
y <- c(86, 19, 18, 170, 43, 20, 40, 11, 5, 28, 4, 3)
tab1 <- matrix(y, byrow=TRUE, nrow = 4)                # Tabelle zu Arbeitslosigkeit
dimnames(tab1) <- list(Ausbildung=c("K","L","F","H"),
                Dauer=c("kurz","mittel","lang"))
mosaicplot(tab1, col=c("gray"), main = "Arbeitslosigkeit")

y <- c(911, 44, 538, 456, 3, 2, 43, 279)
tab2 <- array(y, dim = c(2,2,2))                       # Tabelle zu Drogenmissbrauch
dimnames(tab2) <- list(Zigaretten=c("ja","nein"),
                Marihuana=c("ja","nein"),Alkohol=c("Alkohol: ja","nein"))
mosaicplot(tab2, col="gray", off=c(5,5,5), main="Drogen in der Schule")
```






```
y <- c(86, 19, 18, 170, 43, 20, 40, 11, 5, 28, 4, 3)
n <- sum(y)
tab <- matrix(y, byrow=TRUE, nrow = 4)                 # Tabelle zu Arbeitslosigkeit
dimnames(tab) <- list(ausbildung=c("K","L","F","H"),
                      zeit=c("k","m","l"))
tab
##           zeit
## ausbildung   k  m  l
##          K  86 19 18
##          L 170 43 20
##          F  40 11  5
##          H  28  4  3

zeit.sum <- apply(tab, 2, sum)                         # Randsummen
ausb.sum <- apply(tab, 1, sum)
```

```
L.sat <- -2*sum(y*log(y/n)); L.sat                    # saturiertes Modell
## [1] 1715.89
L.c <- c(0)
for (i in 1:4) { for (j in 1:3) {                     # feste Randsummen
    L.c <- L.c + tab[i, j] * log(ausb.sum[i]*zeit.sum[j]/n^2)  }}

L.c <- -2*L.c;   L.c                                  # beschränktes Modell
##        K
## 1720.577

devianz <- L.c - L.sat; devianz                       # Devianz
##        K
## 4.687199
1-pchisq(devianz, 6)
##         K
## 0.5845111
```

```
chisq.test(tab)
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 4.8195, df = 6, p-value = 0.5672
```

8.5.4 Loglineares Modell am Beispiel von zwei Faktoren

```
y <- c(86, 19, 18, 170, 43, 20, 40, 11, 5, 28, 4, 3)

ausbildung <- c(rep("K",3), rep("L",3), rep("F",3), rep("H",3))
zeit       <- rep(c("k","m","l"),4)
tab        <- data.frame(ausbildung, zeit, y)
fit.sat    <- glm(y ~ zeit + ausbildung + zeit:ausbildung,
                                    family=poisson, data=tab)
fit.c   <- update(fit.sat, . ~ . - zeit:ausbildung)

anova(fit.sat, fit.c)
```

Resid. Df	Resid. Dev	Df	Deviance
0	0.000000	NA	NA
6	4.687199	-6	-4.687199

```
y <- c(86, 19, 18, 170, 43, 20, 40, 11, 5, 28, 4, 3)
ausbildung <- c(rep("K",3), rep("L",3), rep("F",3), rep("H",3))
zeit       <- rep(c("k","m","l"),4)
tab     <- data.frame(ausbildung, zeit, y)
fit     <- glm(y ~ zeit*ausbildung, family=poisson, data=tab)
summary(fit)
##
## Call:
```

```
## glm(formula = y ~ zeit * ausbildung, family = poisson, data = tab)
##
## Deviance Residuals:
##  [1]  0  0  0  0  0  0  0  0  0  0  0  0
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)         3.68888    0.15811  23.331  < 2e-16 ***
## zeitl              -2.07944    0.47434  -4.384 1.17e-05 ***
## zeitm              -1.29098    0.34045  -3.792 0.000149 ***
## ausbildungH        -0.35667    0.24640  -1.448 0.147749
## ausbildungK         0.76547    0.19138   4.000 6.34e-05 ***
## ausbildungL         1.44692    0.17573   8.234  < 2e-16 ***
## zeitl:ausbildungH -0.15415    0.77074  -0.200 0.841479
## zeitm:ausbildungH -0.65493    0.63374  -1.033 0.301401
## zeitl:ausbildungK  0.51547    0.54054   0.954 0.340280
## zeitm:ausbildungK -0.21892    0.42446  -0.516 0.606017
## zeitl:ausbildungL -0.06062    0.52998  -0.114 0.908929
## zeitm:ausbildungL -0.08361    0.38085  -0.220 0.826225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance:  5.0562e+02  on 11  degrees of freedom
## Residual deviance: -2.0650e-14  on  0  degrees of freedom
## AIC: 81.938
##
## Number of Fisher Scoring iterations: 3
```

8.5.5 Dreidimensionale Kontingenztafeln

8.5.5.2 Modellbildung im loglinearen Ansatz

Beispiel Drogenmissbrauch:

```
y <- c(911, 538, 44, 456, 3, 43, 2, 279)
marihuana <- rep(c("ja","nein"), 4)
zigarette <- rep(c("ja","ja","nein","nein"),2)
alkohol   <- c(rep("ja",4), rep("nein",4))
tab       <- data.frame(marihuana, zigarette, alkohol, y)

model <- glm(y ~ marihuana + zigarette + alkohol,
                                family = poisson, data = tab)
summary(model)
##
## Call:
## glm(formula = y ~ marihuana + zigarette + alkohol, family = poisson,
##     data = tab)
##
## Deviance Residuals:
##       1        2        3        4        5        6        7        8
```

```
##  14.522   -7.817  -17.683    3.426  -12.440   -8.436   -8.832   19.639
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)    6.29154    0.03667 171.558  < 2e-16 ***
## marihuananein  0.31542    0.04244   7.431 1.08e-13 ***
## zigarettenein -0.64931    0.04415 -14.707  < 2e-16 ***
## alkoholnein   -1.78511    0.05976 -29.872  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2851.5  on 7  degrees of freedom
## Residual deviance: 1286.0  on 4  degrees of freedom
## AIC: 1343.1
##
## Number of Fisher Scoring iterations: 6
```

```
val        <- matrix(nrow=8, ncol=9)
stats      <- matrix(nrow=9, ncol=4)

fit.a      <- glm(y ~ marihuana + zigarette + alkohol, family=poisson, data=tab)
val[,1]    <- round(fitted.values(fit.a), 1)
stats[1,1] <- round(fit.a$deviance, 1); stats[1,2] <- round(fit.a$aic, 1)
stats[1,3] <- fit.a$df.residual

fit.b1 <- update(fit.a, ~ . + alkohol:zigarette, family=poisson, data=tab)
val[,2] <- round(fitted.values(fit.b1), 1)
stats[2,1] <- round(fit.b1$deviance, 1); stats[2,2] <- round(fit.b1$aic, 1)
stats[2,3] <- fit.b1$df.residual

fit.b2 <- update(fit.a, ~ . + alkohol:marihuana, family=poisson, data=tab)
val[,3] <- round(fitted.values(fit.b2), 1)
stats[3,1] <- round(fit.b2$deviance, 1); stats[3,2] <- round(fit.b2$aic, 1)
stats[3,3] <- fit.b2$df.residual

fit.b3 <- update(fit.a, ~ . + zigarette:marihuana, family=poisson, data=tab)
val[,4] <- round(fitted.values(fit.b3), 1)
stats[4,1] <- round(fit.b3$deviance, 1); stats[4,2] <- round(fit.b3$aic, 1)
stats[4,3] <- fit.b3$df.residual

fit.c1 <- update(fit.a, ~ . + alkohol:marihuana + zigarette:marihuana ,
                 family=poisson, data=tab)
val[,5] <- round(fitted.values(fit.c1), 1)
stats[5,1] <- round(fit.c1$deviance, 1); stats[5,2] <- round(fit.c1$aic, 1)
stats[5,3] <- fit.c1$df.residual

fit.c2 <- update(fit.a, ~ . + alkohol:marihuana + alkohol:zigarette ,
                 family=poisson, data=tab)
val[,6] <- round(fitted.values(fit.c2), 1)
stats[6,1] <- round(fit.c2$deviance, 1); stats[6,2] <- round(fit.c2$aic, 1)
stats[6,3] <- fit.c2$df.residual
```

```
fit.c3 <- update(fit.a, ~ . + alkohol:zigarette + zigarette:marihuana ,
                 family=poisson, data=tab)
val[,7] <- round(fitted.values(fit.c3), 1)
stats[7,1] <- round(fit.c3$deviance, 1); stats[7,2] <- round(fit.c3$aic, 1)
stats[7,3] <- fit.c3$df.residual

fit.d <- update(fit.a, ~ . + alkohol:zigarette + zigarette:marihuana
                + alkohol:marihuana, family=poisson, data=tab)
val[,8] <- round(fitted.values(fit.d), 1)
stats[8,1] <- round(fit.d$deviance, 1); stats[8,2] <- round(fit.d$aic, 1)
stats[8,3] <- fit.d$df.residual

fit.0 <- glm(y ~ marihuana*zigarette*alkohol, family=poisson, data=tab)
val[,9] <- round(fitted.values(fit.0), 1)
stats[9,1] <- round(fit.0$deviance, 1); stats[9,2] <- round(fit.0$aic, 1)
stats[9,3] <- fit.0$df.residual

for (i in 1:9) {stats[i,4] <- round(pchisq(stats[i,1],
                stats[i,3], lower.tail = FALSE), 10)}
dimnames(stats) <- list(Modell=c("A","B1","B2","B3","C1","C2","C3","D","0"),
                Statistik=c("Devianz","AIC","Freiheitsgrad","P-Wert"))
stats
```

```
##       Statistik
## Modell Devianz    AIC Freiheitsgrad    P-Wert
##     A   1286.0 1343.1             4 0.0000000
##     B1   843.8  902.9             3 0.0000000
##     B2   939.6  998.6             3 0.0000000
##     B3   534.2  593.3             3 0.0000000
##     C1   187.8  248.8             2 0.0000000
##     C2   497.4  558.4             2 0.0000000
##     C3    92.0  153.1             2 0.0000000
##     D      0.4   63.4             1 0.5270893
##     0      0.0   65.0             0 1.0000000
```

Tabelle zum Beispiel:

```
chi <- rep(NA, 9)

for (i in 1:8) {chi[i] <- sum((val[,9] - val[,i])^2 / val[,i])}
val <- rbind(val, chi)
dimnames(val) <- list(Feld=c(1,2,3,4,5,6,7,8,"chi"),
                      Modell=c("A","B1","B2","B3","C1","C2","C3","D","0"))
as.data.frame(round(val, 1))
```

	A	B1	B2	B3	C1	C2	C3	D	0
1	540.0	611.2	627.3	782.7	909.2	710.0	885.9	910.4	911
2	740.2	837.8	652.9	497.5	438.8	739.0	563.1	538.6	538
3	282.1	210.9	327.7	39.4	45.8	245.0	29.4	44.6	44
4	386.7	289.1	341.1	629.4	555.2	255.0	470.6	455.4	456
5	90.6	19.4	3.3	131.3	4.8	0.7	28.1	3.6	3
6	124.2	26.6	211.5	83.5	142.2	45.3	17.9	42.4	43
7	47.3	118.5	1.7	6.6	0.2	4.3	16.6	1.4	2
8	64.9	162.5	110.5	105.6	179.8	276.7	264.4	279.6	279
chi	1411.0	704.8	824.1	505.6	181.0	443.8	80.8	0.4	NA

Variablenauswahl mit Fuktion stepAIC() in library(MASS)

```
library(MASS)

y <- c(911, 538, 44, 456, 3, 43, 2, 279)
marihuana <- rep(c("ja","nein"), 4)
zigarette <- rep(c("ja","ja","nein","nein"),2)
alkohol   <- c(rep("ja",4), rep("nein",4))
tab       <- data.frame(marihuana, zigarette, alkohol, y)

model.step <- stepAIC(model, list(upper = ~ .^3,
                      lower = formula(model)), trace=FALSE)
model.step$anova
```

Step	Df	Deviance	Resid. Df	Resid. Dev	AIC
	NA	NA	4	1286.0199544	1343.06338
+ marihuana:zigarette	1	751.80828	3	534.2116714	593.25510
+ zigarette:alkohol	1	442.19331	2	92.0183606	153.06179
+ marihuana:alkohol	1	91.64437	1	0.3739859	63.41741

Interpretation der Modellparameter:

```
y <- c(911, 538, 44, 456, 3, 43, 2, 279)
m <- rep(c("ja","nein"), 4)
z <- rep(c("ja","ja","nein","nein"),2)
a   <- c(rep("ja",4), rep("nein",4))
tab <- data.frame(m, z, a, y)

fit <- glm(y ~ m * a * z - m:z:a, family=poisson, data=tab, x=T)
summary(fit)
##
## Call:
## glm(formula = y ~ m * a * z - m:z:a, family = poisson, data = tab,
##     x = T)
##
## Deviance Residuals:
##       1        2        3        4        5        6        7        8
##  0.02044  -0.02658  -0.09256   0.02890  -0.33428   0.09452   0.49134  -0.03690
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  6.81387    0.03313 205.699  < 2e-16 ***
## mnein       -0.52486    0.05428  -9.669  < 2e-16 ***
## anein       -5.52827    0.45221 -12.225  < 2e-16 ***
## znein       -3.01575    0.15162 -19.891  < 2e-16 ***
## mnein:anein  2.98601    0.46468   6.426 1.31e-10 ***
## mnein:znein  2.84789    0.16384  17.382  < 2e-16 ***
## anein:znein  2.05453    0.17406  11.803  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 2851.46098  on 7  degrees of freedom
## Residual deviance:    0.37399  on 1  degrees of freedom
## AIC: 63.417
##
## Number of Fisher Scoring iterations: 4

fit$x
##   (Intercept) mnein anein znein mnein:anein mnein:znein anein:znein
## 1           1     0     0     0           0           0           0
## 2           1     1     0     0           0           0           0
## 3           1     0     0     1           0           0           0
## 4           1     1     0     1           0           1           0
## 5           1     0     1     0           0           0           0
## 6           1     1     1     0           1           0           0
## 7           1     0     1     1           0           0           1
## 8           1     1     1     1           1           1           1
## attr(,"assign")
## [1] 0 1 2 3 4 5 6
## attr(,"contrasts")
## attr(,"contrasts")$m
## [1] "contr.treatment"
```

```
##
## attr(,"contrasts")$a
## [1] "contr.treatment"
##
## attr(,"contrasts")$z
## [1] "contr.treatment"
```

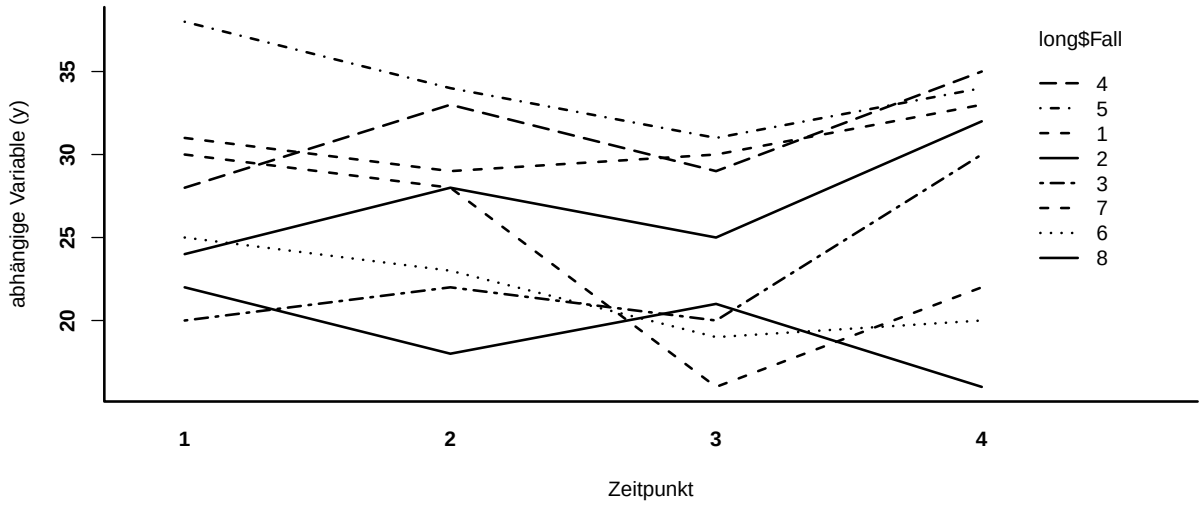
8.6 Modelle zu wiederholten Messungen

Format der Daten (long vs. wide):

```
wide <- read.csv2("lmebroad.csv")                    # weites Format
wide
```

Fall	Gruppe	y1	y2	y3	y4
1	1	31	29	30	33
2	1	24	28	25	32
3	1	20	22	20	30
4	1	28	33	29	35
5	2	38	34	31	34
6	2	25	23	19	20
7	2	30	28	16	22
8	2	22	18	21	16

```
long <- reshape(wide, direction="long",                    # langes Format
                idvar = "Fall", timevar = "Zeit", v.names="y", varying=list(3:6))
attach(long)

par(lwd=2, font.axis=2, bty="l", ps=12, mfrow=c(1,1))
interaction.plot(long$Zeit, long$Fall, long$y,
            ylab = "abhängige Variable (y)",lwd=2, xlab = "Zeitpunkt")
```



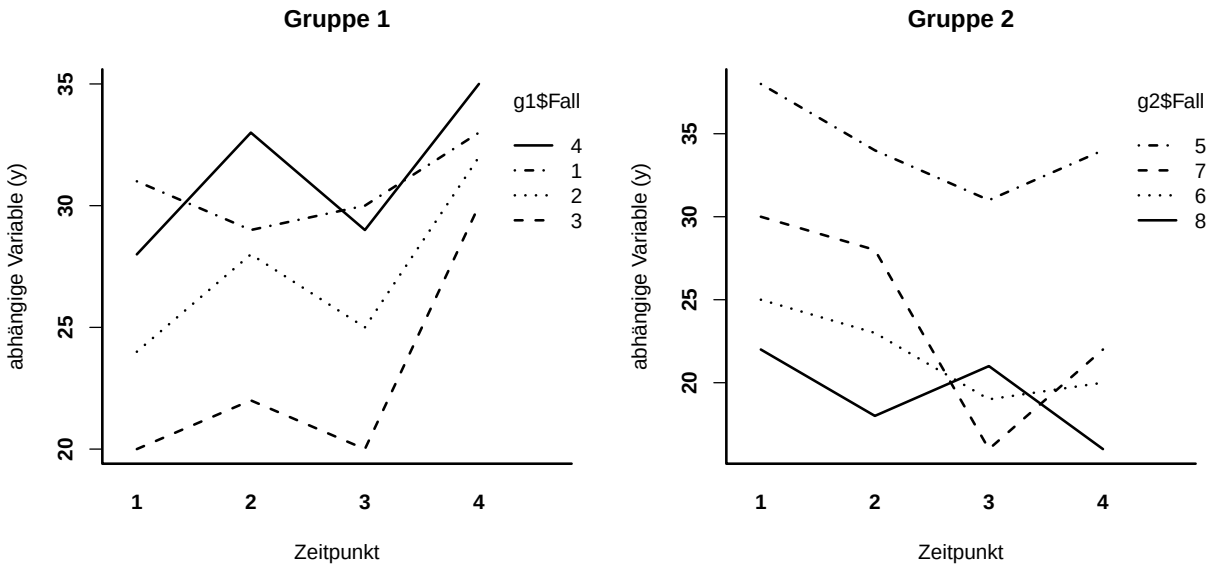
ANOVA ohne Berücksichtugung der Gruppe:

```
long.aov1 <- aov(long$y ~ as.factor(long$Zeit) + Error(factor(long$Fall)))

summary(long.aov1)
##
## Error: factor(long$Fall)
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  7  779.4   111.3
##
## Error: Within
##                       Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(long$Zeit)  3  73.12   24.38   1.929  0.156
## Residuals            21 265.38   12.64
```

Anova mit Berücksichtigung der Gruppe:

```
long.aov2 <- aov(long$y ~ factor(long$Gruppe) * factor(long$Zeit)
                 + Error(factor(Fall)))

summary(long.aov2)
##
## Error: factor(Fall)
##                      Df Sum Sq Mean Sq F value Pr(>F)
## factor(long$Gruppe)  1   84.5    84.5    0.73  0.426
## Residuals            6  694.9   115.8
##
## Error: Within
##                                      Df Sum Sq Mean Sq F value   Pr(>F)
## factor(long$Zeit)                     3  73.12   24.37   4.174 0.020804 *
## factor(long$Gruppe):factor(long$Zeit)  3 160.25   53.42   9.146 0.000679 ***
## Residuals                            18 105.12    5.84
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
g1 <- long[Gruppe==1,]; g2 <- long[Gruppe==2,]

par(lwd=2, font.axis=2, bty="l", ps=12, mfrow=c(1,2))
interaction.plot(g1$Zeit, g1$Fall, g1$y, ylab = "abhängige Variable (y)",
                 lwd=2, main="Gruppe 1", xlab = "Zeitpunkt")

interaction.plot(g2$Zeit, g2$Fall, g2$y, ylab = "abhängige Variable (y)",
                 lwd=2, main="Gruppe 2", xlab = "Zeitpunkt")
```


Gruppe 1
abhängige Variable (y)
20
25
30
35
1
2
3
4
Zeitpunkt
g1$Fall
4
1
2
3
Gruppe 2
abhängige Variable (y)
20
25
30
35
1
2
3
4
Zeitpunkt
g2$Fall
5
7
6
8

8.6.2 Lineare gemischte Modelle

```
wide <- read.csv2("lmebroad.csv")                          # weites Format
wide
```

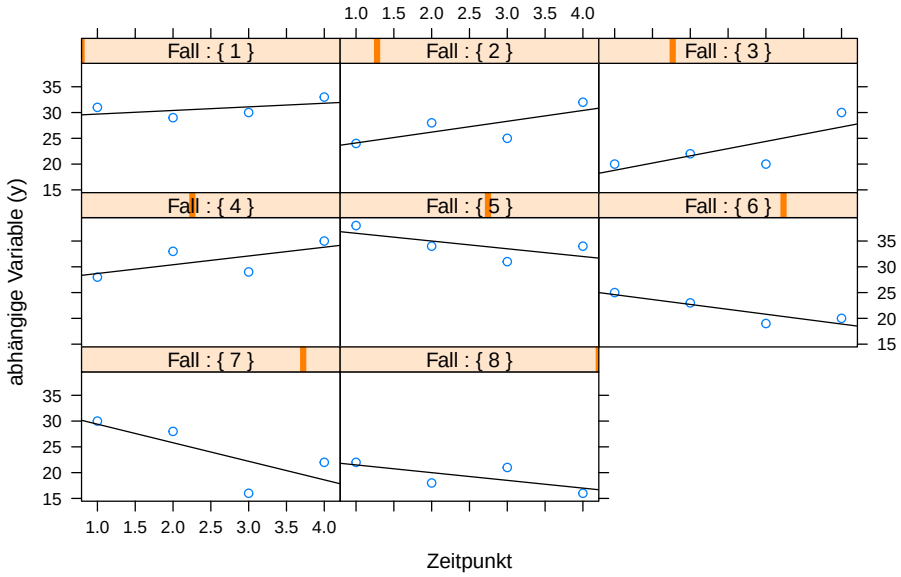
Fall	Gruppe	y1	y2	y3	y4
1	1	31	29	30	33
2	1	24	28	25	32
3	1	20	22	20	30
4	1	28	33	29	35
5	2	38	34	31	34
6	2	25	23	19	20
7	2	30	28	16	22
8	2	22	18	21	16

```
long <- reshape(wide, direction="long",                          # langes Format
                idvar = "Fall", timevar = "Zeit", v.names="y", varying=list(3:6))
attach(long)

int  <- by(long, Fall,
        function(data) coefficients(lm(y ~ Zeit, data = data))[[1]])
rate <- by(long, Fall,
        function(data) coefficients(lm(y ~ Zeit, data = data))[[2]])
r <- as.data.frame(cbind(wide$Fall, wide$Gruppe,
        unlist(int), unlist(rate)))
colnames(r) <- c("i", "Gruppe", "beta_0", "beta_1")
r
```

i	Gruppe	beta_0	beta_1
1	1	29.0	0.7
2	1	22.0	2.1
3	1	16.0	2.8
4	1	27.0	1.7
5	2	38.0	-1.5
6	2	26.5	-1.9
7	2	33.0	-3.6
8	2	23.0	-1.5

Lattice-Plot mit Funktion xyplot() in library(lattice)

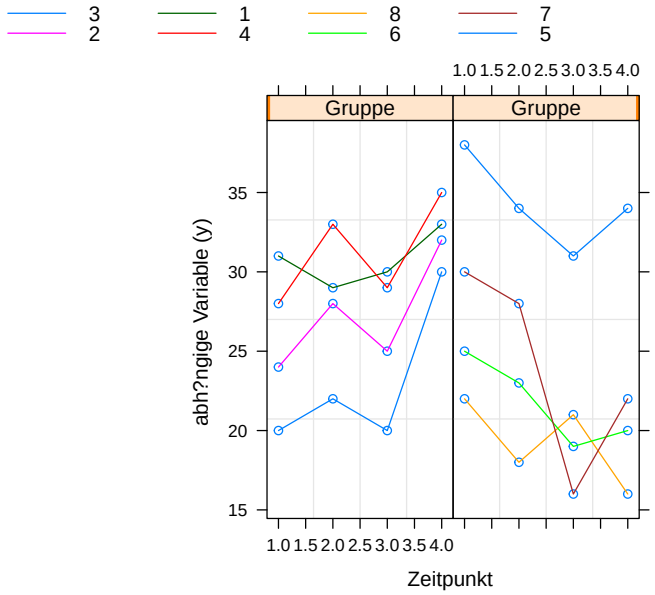
```
library(lattice); par(cex=2)

xyplot(y ~ Zeit | Fall, data=long, aspect=1/2,
       xlab="Zeitpunkt", ylab="abhängige Variable (y)",
       lattice.options = list(lwd=3),
       strip = strip.custom(strip.names = TRUE, strip.levels = TRUE),
       panel = function(x, y, subscripts){
               panel.xyplot(x, y)
               panel.lmline(x, y)}, as.table=T)
```



Funktion groupedData() in library(nlme)

```
library(nlme)

long.new <- groupedData( y ~ Zeit | Fall, data = long, outer = ~ as.factor(Gruppe),
          labels = list( x = "Zeitpunkt", y = "abh?ngige Variable (y)" ))

long.lis <- lmList(y ~ Zeit | Fall, data=long.new); long.lis
## Call:
##   Model: y ~ Zeit | Fall
##    Data: long.new
##
## Coefficients:
##   (Intercept) Zeit
## 3        16.0  2.8
## 2        22.0  2.1
## 1        29.0  0.7
## 4        27.0  1.7
## 8        23.0 -1.5
## 6        26.5 -1.9
```

```
## 7        33.0 -3.6
## 5        38.0 -1.5
##
## Degrees of freedom: 32 total; 16 residual
## Residual standard error: 3.112475

plot(long.new, outer = ~Gruppe)                    # trellis plot by Subject
```




```
predict(long.lis, se.fit = TRUE)[1:10,]
```

Fall	fit	se.fit
3	18.8	2.604083
3	21.6	1.704773
3	24.4	1.704773
3	27.2	2.604083
2	24.1	2.604083
2	26.2	1.704773
2	28.3	1.704773
2	30.4	2.604083
1	29.7	2.604083
1	30.4	1.704773

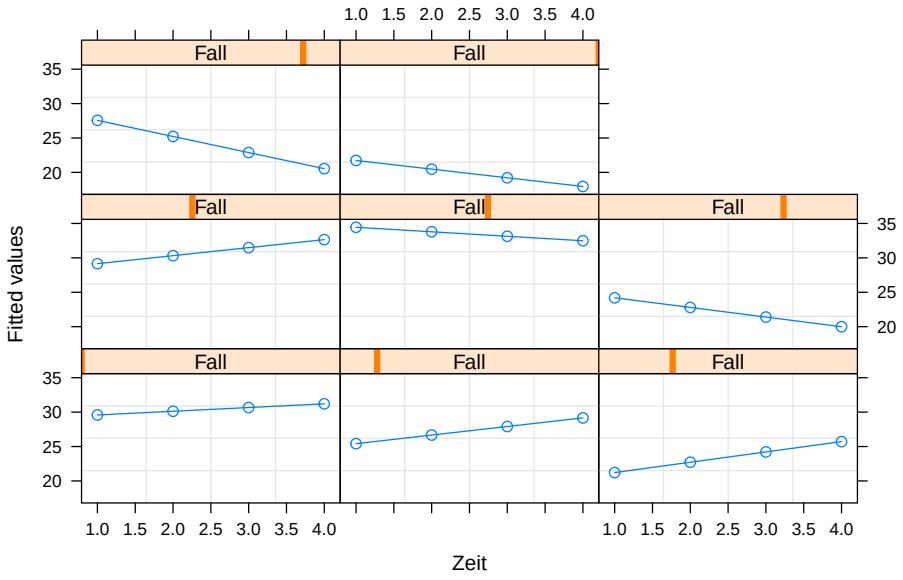
```
lme(long.lis)
## Linear mixed-effects model fit by REML
##   Data: long.new
##   Log-restricted-likelihood: -92.01803
##   Fixed: y ~ Zeit
## (Intercept)         Zeit
##     26.8125     -0.1500
```

```
##
## Random effects:
##  Formula: ~Zeit | Fall
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev   Corr
## (Intercept) 5.622040 (Intr)
## Zeit        1.810983 -0.524
## Residual    3.112470
##
## Number of Observations: 32
## Number of Groups: 8
```

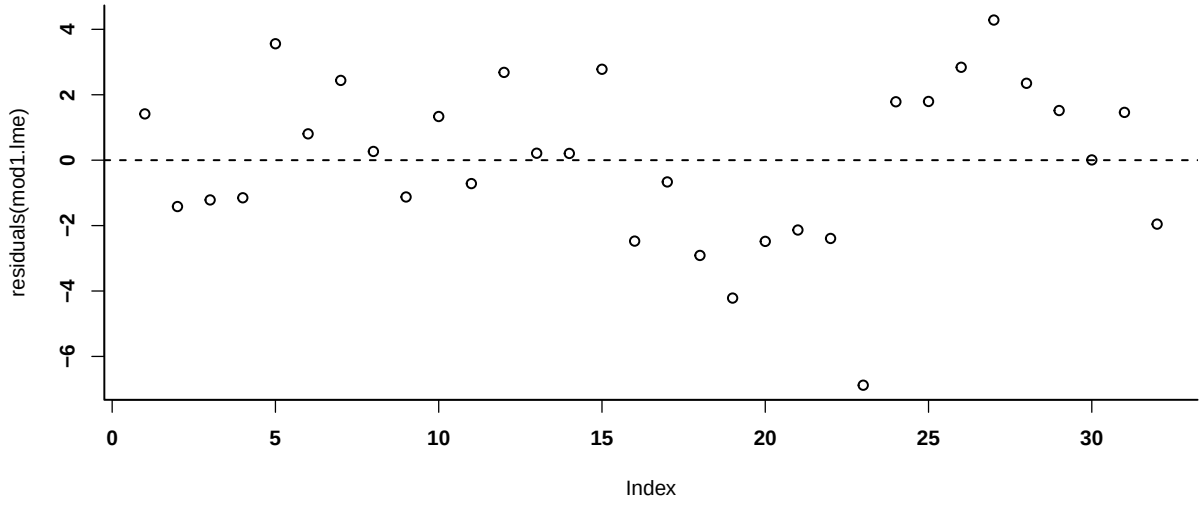
Modell 1 ohne Berücksichtigung der Gruppe:

```
mod1.lme <- lme( y ~ Zeit , method="REML", data = long,  random = ~ Zeit | Fall)

summary(mod1.lme)
## Linear mixed-effects model fit by REML
##  Data: long
##        AIC      BIC    logLik
##   196.0361 204.4433 -92.01803
##
## Random effects:
##  Formula: ~Zeit | Fall
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev   Corr
## (Intercept) 5.622037 (Intr)
## Zeit        1.810984 -0.524
## Residual    3.112472
##
## Fixed effects: y ~ Zeit
##               Value Std.Error DF   t-value p-value
## (Intercept) 26.8125 2.4015235 23 11.164788  0.0000
## Zeit        -0.1500 0.8075548 23 -0.185746  0.8543
##  Correlation:
##      (Intr)
## Zeit -0.656
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.21042915 -0.49869103  0.06716016  0.57434440  1.37636691
##
## Number of Observations: 32
## Number of Groups: 8
```

```
par(lwd=1.5, font.axis=2, bty="l", ps=12, mfrow=c(1,1))
plot(mod1.lme, fitted(.) ~ Zeit | Fall, aspect=1/2, type="b")
```



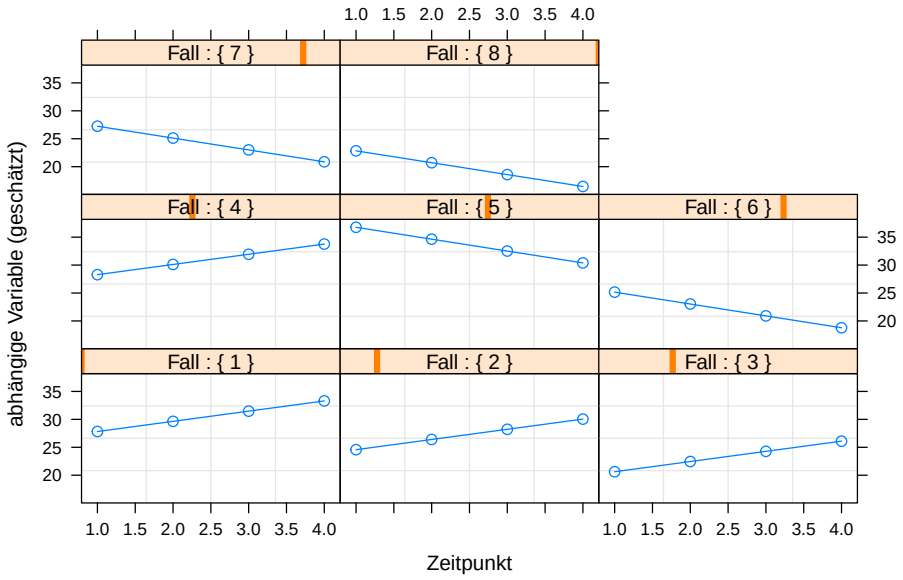

```
par(lwd=1.5, font.axis=2, bty="l", ps=12, mfrow=c(1,1))
plot(residuals(mod1.lme))
abline(h=0, lty=2)
```



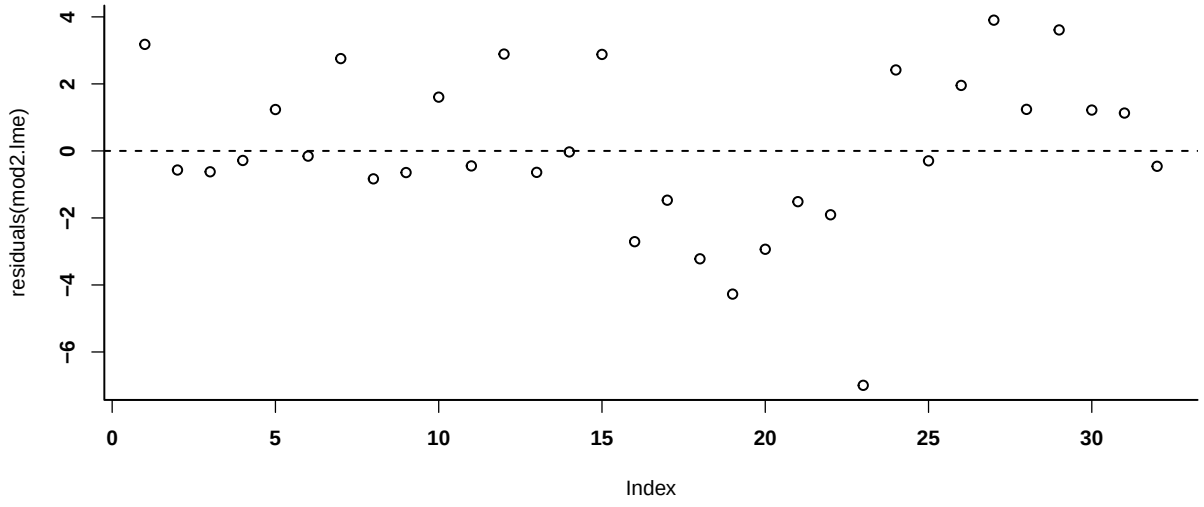
Modell 2 mit Berücksichtigung der Gruppe:

```
mod2.lme <- lme( y ~ Zeit*as.factor(Gruppe) , method="REML",
                                  data = long, random = ~ Zeit | Fall)
summary(mod2.lme)
## Linear mixed-effects model fit by REML
##  Data: long
##        AIC      BIC    logLik
##   181.9428 192.6005 -82.97141
##
## Random effects:
##  Formula: ~Zeit | Fall
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev       Corr
## (Intercept) 5.1855360447 (Intr)
## Zeit        0.0004573108 -0.001
## Residual    2.8728746222
##
## Fixed effects: y ~ Zeit * as.factor(Gruppe)
##                           Value Std.Error DF   t-value p-value
## (Intercept)              23.500  3.133285 22  7.500116  0.0000
## Zeit                      1.825  0.642394 22  2.840934  0.0095
## as.factor(Gruppe)2        6.625  4.431134  6  1.495103  0.1855
## Zeit:as.factor(Gruppe)2 -3.950  0.908483 22 -4.347908  0.0003
##  Correlation:
##                          (Intr) Zeit   a.(G)2
## Zeit                     -0.513
## as.factor(Gruppe)2       -0.707  0.362
## Zeit:as.factor(Gruppe)2  0.362 -0.707 -0.513
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.4349838 -0.3457088 -0.1011733  0.5889909  1.3581237
##
## Number of Observations: 32
## Number of Groups: 8
```

```
par(lwd=1.5, font.axis=2, bty="l", ps=12, mfrow=c(1,1))
plot(mod2.lme, fitted(.) ~ Zeit | Fall, aspect=1/2, type="b",
     strip = strip.custom(strip.names = TRUE, strip.levels = TRUE),
     ylab="abhängige Variable (geschätzt)", xlab="Zeitpunkt")
```

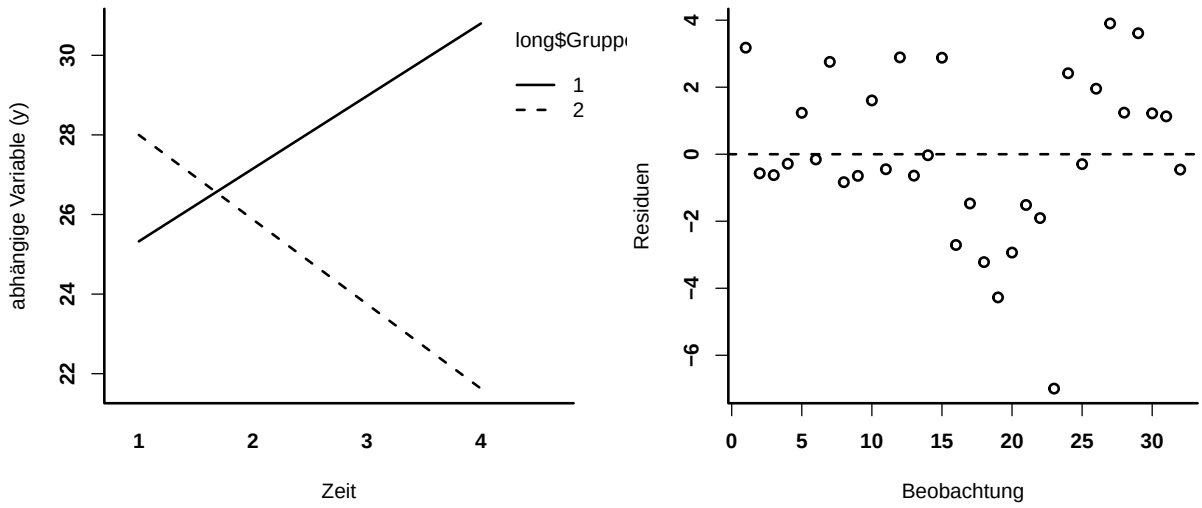



```
par(lwd=1.5, font.axis=2, bty="l", ps=12, mfrow=c(1,1))
plot(residuals(mod2.lme)); abline(h=0, lty=2)
```



Residuenplot zu Modell 2:

```
par(lwd=2, font.axis=2, bty="l", ps=12, mfrow=c(1,2))
fit <- unlist(fitted.values(mod2.lme))
interaction.plot(long$Zeit, long$Gruppe, fit, xlab="Zeit", lwd=2,
                          ylab="abhängige Variable (y)", lty=1:2)
plot(residuals(mod2.lme), ylab="Residuen", xlab="Beobachtung")
abline(h=0, lty=2)
```



Vergleich zweier Modelle:

```
mod2.lme  <- lme( y ~ Zeit * as.factor(Gruppe) , method="ML",
                  data = long, random = ~ Zeit | Fall)
mod2a.lme <- lme( y ~ Zeit * as.factor(Gruppe) , method="ML",
                  data = long, random = ~ 1 | Fall)
tab <- summary(anova(mod2.lme, mod2a.lme))
tab[,3:9]
##        df           AIC             BIC           logLik           Test
##  Min.   :6.0   Min.   :187.1   Min.   :195.9   Min.   :-87.55         :1
##  1st Qu.:6.5   1st Qu.:188.1   1st Qu.:197.6   1st Qu.:-87.55   1 vs 2:1
##  Median :7.0   Median :189.1   Median :199.4   Median :-87.55
##  Mean   :7.0   Mean   :189.1   Mean   :199.4   Mean   :-87.55
##  3rd Qu.:7.5   3rd Qu.:190.1   3rd Qu.:201.1   3rd Qu.:-87.55
##  Max.   :8.0   Max.   :191.1   Max.   :202.8   Max.   :-87.55
##
##     L.Ratio     p-value
##  Min.   :0   Min.   :1
##  1st Qu.:0   1st Qu.:1
##  Median :0   Median :1
##  Mean   :0   Mean   :1
##  3rd Qu.:0   3rd Qu.:1
##  Max.   :0   Max.   :1
##  NA's   :1   NA's   :1
```

8.6.3 Analyse von Clusterdaten

Clustersampling:

```
cluster.smpl <- function(df, id, k = 1) {       # df  Dataframe
                                                # id  Spalte mit Identifikation
                                                # k   Anzahl der Fälle
do.call("rbind", lapply(split(df, df[,id]),
        function(x) { x[sample(seq_len(nrow(x)), size=k), ] } ))
                                        }

sort.data <- function(df, col) {                # Sortieren der Daten
    sd <- do.call("order", as.data.frame(df[ ,col]))
    df[sd, ]
                                        }
```

```
dat <- read.csv2("cluster.csv")

sample <- cluster.smpl(dat, 2, 1); sample
```

nro	familie	fall	geschlecht	groesse
1	1	1	f	170.18
6	2	1	f	160.02
14	3	4	m	165.12
18	4	3	m	170.18

```
sort.data(sample, 5)                            # Sort nach Körpergröße
```

	nro	familie	fall	geschlecht	groesse
2	6	2	1	f	160.02
3	14	3	4	m	165.12
1	1	1	1	f	170.18
4	18	4	3	m	170.18

Beispiel Körpergrößen in Familien:

```
height <- read.csv2("cluster.csv")

height <- sort.data(height, 2)                  # Sort nach Familien

                                                # Weites Format!

wheight <- reshape(height, v.names=c("geschlecht", "groesse"),
                   idvar="familie", timevar="fall", direction="wide")
wheight[,1:7]
```

	nro	familie	geschlecht.1	groesse.1	geschlecht.2	groesse.2	geschlecht.3
1	1	1	f	170.18	f	167.64	f
6	6	2	f	160.02	f	165.44	f
11	11	3	f	154.94	f	157.48	m
16	16	4	f	170.18	f	167.64	m

```
attach(wheight)
                                                  # Korrelations- bzw. Kovarianzmatrix
km <-cor(cbind(groesse.1, groesse.2, groesse.3, groesse.4, groesse.5))
round(km, 4)
##           groesse.1 groesse.2 groesse.3 groesse.4 groesse.5
## groesse.1    1.0000    0.8947   -0.3143    0.7331    0.8912
## groesse.2    0.8947    1.0000   -0.1074    0.8663    0.9142
## groesse.3   -0.3143   -0.1074    1.0000   -0.4720   -0.4994
## groesse.4    0.7331    0.8663   -0.4720    1.0000    0.9584
## groesse.5    0.8912    0.9142   -0.4994    0.9584    1.0000
```

Ansatz mit Varianzanalyse:

```
var.tab <- summary(aov(groesse ~ as.factor(familie), height))
var.tab
##                    Df Sum Sq Mean Sq F value Pr(>F)
## as.factor(familie)  3  367.2   122.4   3.327 0.0464 *
## Residuals          16  588.8    36.8
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

msq <- var.tab[[1]][,3]; BMS <- msq[1];  WMS <- msq[2];  m <- 5

ICC <- (BMS - WMS)/(BMS + (m-1) * WMS)      # Intraklassen-Korrelationskoeffizien
ICC
## [1] 0.3175518

VIF <- 1 + (m-1) * ICC                      # Varianzinflationsfaktor
VIF
## [1] 2.270207
```

Funktion lme() in library(nlme)

```
library(nlme)
summary(lme(groesse ~ geschlecht, data = height,
                      random = ~ 1 | familie, method="REML"))
## Linear mixed-effects model fit by REML
##  Data: height
##        AIC      BIC    logLik
##   117.1171 120.6786 -54.55855
##
## Random effects:
##  Formula: ~1 | familie
##         (Intercept) Residual
## StdDev:    5.311524 3.591535
```

```
##
## Fixed effects: groesse ~ geschlecht
##                 Value Std.Error DF  t-value p-value
## (Intercept) 164.06312  2.892666 15 56.71691   0e+00
## geschlechtm   8.96777  1.636450 15  5.48001   1e-04
##  Correlation:
##             (Intr)
## geschlechtm -0.283
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.7118412 -0.5278310  0.1415506  0.5221704  1.4489894
##
## Number of Observations: 20
## Number of Groups: 4
```

t-Test (ohne Berücksichtigung der Cluster-Struktur):

```
t.test(groesse ~ geschlecht, height)
##
##  Welch Two Sample t-test
##
## data:  groesse by geschlecht
## t = -2.9213, df = 17.696, p-value = 0.009234
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -13.488681  -2.195319
## sample estimates:
## mean in group f mean in group m
##         164.626         172.468
```

ICC und VIF aus dichotomen Daten:

```
VIF.ICC <- function(x, m) {            # ICC und VIF aus dichotomen Daten
    n   <- sum(m); k <- length(m)
    mm  <- mean(m)
    m0  <- mm - sum((m - mm)**2) / (k * (k - 1) * mm)
    p <- x/m
    pm  <- sum(x)/sum(m)
    BMS <- sum((x - m * pm)**2 / m) / k
    WMS <- sum( x * (m - x)  / m) / (k * (pm - 1))
    ICC <- (BMS - WMS) / (BMS - (m0 - 1)*WMS)
    VIF <- 1 + (m0-1)*ICC
    cat("ICC = ", round(ICC,4), "sowie VIF = ", round(VIF,4),"\n")
          }
```

Beispiel Toxizitätsstudie (Laktation):

```
surv.c <- matrix(c(13,12, 9, 9, 8, 8,12,11, 9, 9, 8,11, 4, 5, 7, 7,
                   0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 1, 2, 3, 3),
            nrow=2, byrow=TRUE, dimnames = list(c("ja","nein"),1:16))
```

```
surv.t <- matrix(c(12,11,10, 9,10, 9, 9, 8, 8, 4, 7, 4, 5, 3, 3, 0,
                    0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 2, 3, 5, 3, 7, 7),
           nrow=2, byrow=TRUE, dimnames = list(c("ja","nein"),1:16))

                                        # mittlerer Laktationsindex je Wurf
LIc <- surv.c[1,]/apply(surv.c, 2, sum); mean(LIc)
## [1] 0.893067
LIt <- surv.t[1,]/apply(surv.t, 2, sum); mean(LIt)
## [1] 0.7460047

                                        # ICC und VIF zum Laktationsindex
VIF.ICC(surv.c[2,], apply(surv.c, 2, sum))
## ICC =  0.1253 sowie VIF =  2.1075
VIF.ICC(surv.t[2,], apply(surv.t, 2, sum))
## ICC =  0.1825 sowie VIF =  2.4667
```

Erzeuge Daten im 'langen' Format zu jedem Wurf:

```
i.c  <- apply(surv.c, 2, sum); n.c <- sum(i.c)
cdat <- as.data.frame(matrix(rep(0, 4*n.c), nrow=n.c, byrow=TRUE))
colnames(cdat) <- c("Gruppe","Wurf","Tier","Status"); cnt  <- 1
for (i in 1:16) {
    if (surv.c[1,i]>0)
    for (k in 1:surv.c[1,i])
            {cdat[cnt,] <- c("Kontrolle", i, k, 0); cnt <- cnt + 1 }
    if (surv.c[2,i]>0)
    for (k in 1:surv.c[2,i])
            {cdat[cnt,] <- c("Kontrolle", i, k, 1); cnt <- cnt + 1 }
    }
i.t  <- apply(surv.t, 2, sum); n.t <- sum(i.t)
tdat <- as.data.frame(matrix(rep(NA, 4*n.t), nrow=n.t, byrow=TRUE))
colnames(tdat) <- c("Gruppe","Wurf","Tier","Status"); cnt  <- 0
for (i in 1:16) {
    if (surv.t[1,i]>0)
    for (k in 1:surv.t[1,i])
            {cnt <- cnt + 1; tdat[cnt,] <- c("Substanz", i+16, k, 0)}
    if (surv.t[2,i]>0)
    for (k in 1:surv.t[2,i])
            {cnt <- cnt + 1; tdat[cnt,] <- c("Substanz", i+16, k, 1)}
    }

Sdat        <- rbind(cdat, tdat)
Sdat$Gruppe <- as.factor(Sdat$Gruppe)
Sdat$Wurf   <- as.numeric(Sdat$Wurf)
Sdat$Tier   <- as.numeric(Sdat$Tier)
Sdat$Status <- as.numeric(Sdat$Status)

# write.csv2(Sdat, file="Teratogen Beispiel.csv")
```

Verallgemeinertes lineares gemischtes Modell für Cluster-Daten:

Funktion glmmPQL() in library(MASS)

```
Sdat  <- read.csv2(file="teratogen.csv")
attach(Sdat); tab <- table(Gruppe, Status); tab
##            Status
## Gruppe        0   1
##   Kontrolle 142  16
##   Substanz  112  33

library(MASS)
summary(glmmPQL(Status ~ Gruppe, random = ~ 1 | Wurf,
                                 family = binomial, data = Sdat))
## Linear mixed-effects model fit by maximum likelihood
##  Data: Sdat
##   AIC BIC logLik
##    NA  NA     NA
##
## Random effects:
##  Formula: ~1 | Wurf
##         (Intercept)  Residual
## StdDev:    1.268235 0.8478978
##
## Variance function:
##  Structure: fixed weights
##  Formula: ~invwt
## Fixed effects: Status ~ Gruppe
##                     Value Std.Error  DF   t-value p-value
## (Intercept)    -2.3732964 0.4089413 271 -5.803514  0.0000
## GruppeSubstanz  0.9620611 0.5590925  30  1.720755  0.0956
##  Correlation:
##                (Intr)
## GruppeSubstanz -0.731
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -1.4931947 -0.4538231 -0.3466446 -0.2406653  4.0124899
##
## Number of Observations: 303
## Number of Groups: 32
```

8.6.4 Verallgemeinerte Schätzgleichungen

Beispiel 1 Körpergröße mit Funktion geese() in library(geepack):

```
height <- read.csv2("cluster.csv")
height <- sort.data(height, 2)

library(geepack)
summary(geese(groesse ~ geschlecht, id = familie, data=height,
                            corstr="exchangeable", family=gaussian))
##
## Call:
## geese(formula = groesse ~ geschlecht, id = familie, data = height,
##     family = gaussian, corstr = "exchangeable")
##
## Mean Model:
##  Mean Link:                 identity
##  Variance to Mean Relation: gaussian
##
##  Coefficients:
##               estimate   san.se       wald            p
## (Intercept) 164.076614 2.290952 5129.34008 0.000000e+00
## geschlechtm   8.940771 1.890909   22.35676 2.264149e-06
##
## Scale Model:
##  Scale Link:                identity
##
##  Estimated Scale Parameters:
##             estimate   san.se     wald           p
## (Intercept) 32.72887 11.55003 8.029638 0.004601804
##
## Correlation Model:
##  Correlation Structure:     exchangeable
##  Correlation Link:          identity
##
##  Estimated Correlation Parameters:
##        estimate   san.se    wald            p
## alpha 0.6303665 0.181695 12.0365 0.0005216888
##
## Returned Error Value:    0
## Number of clusters:   4   Maximum cluster size: 5
```

Beispiel 2 Toxizitätsstudie mit Funktion geese() in library(geepack):

```
Sdat  <- read.csv2(file="teratogen.csv")
attach(Sdat)

library(geepack)
summary(geese(Status ~ Gruppe, id = Wurf, data=Sdat,
                            corstr="exchangeable", family=binomial))
##
## Call:
## geese(formula = Status ~ Gruppe, id = Wurf, data = Sdat, family = binomial,
```

```
##     corstr = "exchangeable")
##
## Mean Model:
##  Mean Link:                 logit
##  Variance to Mean Relation: binomial
##
##  Coefficients:
##                 estimate    san.se      wald            p
## (Intercept)    -2.148397 0.2839705 57.237760 3.863576e-14
## GruppeSubstanz  1.014390 0.4714840  4.628885 3.143798e-02
##
## Scale Model:
##  Scale Link:                identity
##
##  Estimated Scale Parameters:
##             estimate    san.se     wald            p
## (Intercept) 0.9647091 0.2772236 12.10967 0.0005016099
##
## Correlation Model:
##  Correlation Structure:     exchangeable
##  Correlation Link:          identity
##
##  Estimated Correlation Parameters:
##       estimate     san.se     wald          p
## alpha 0.1555769 0.07919263 3.859406 0.04946784
##
## Returned Error Value:    0
## Number of clusters:   32   Maximum cluster size: 13
```

8.7 Analyse von Überlebenszeiten

8.7.1 Kaplan-Meier Schätzung der Überlebensfuktion

Beispiel Überleben nach Chemotherapie mit Funktion survfit() in library(survival):

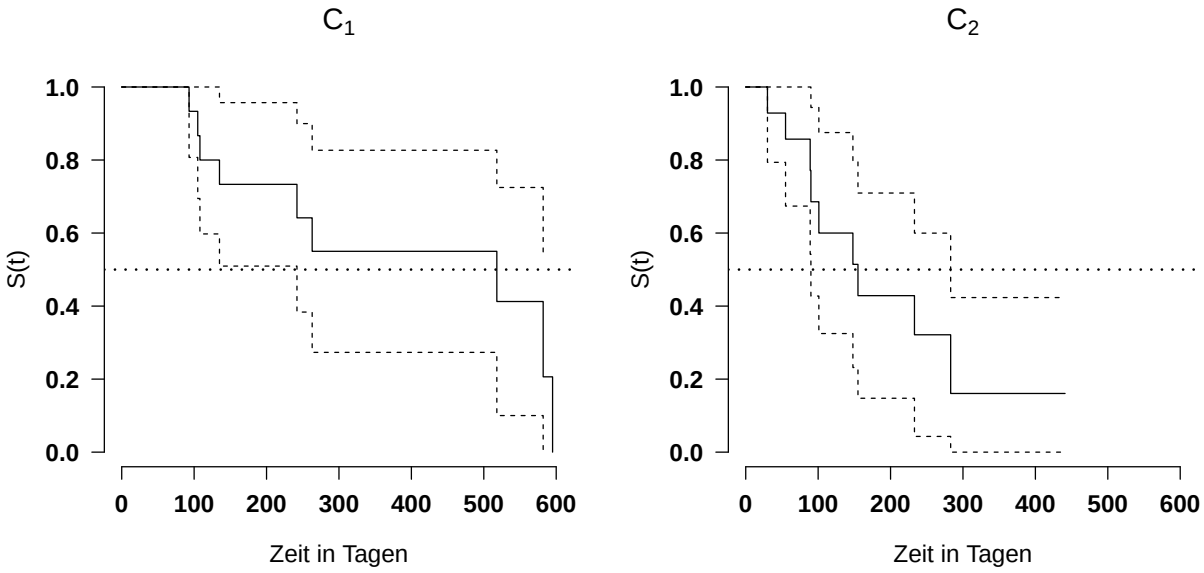
```
library(survival)
library(muhaz)

chemo <- read.csv2("chemotherapie.csv")
attach(chemo)
chemo[1:10,]
```

X	gruppe	zeit	status
1	1	26	0
2	1	50	0
3	1	51	0
4	1	57	0
5	1	70	0
6	1	93	1
7	1	105	1
8	1	108	1
9	1	135	1
10	1	193	0

```
tab <- summary(survfit(Surv(zeit, status) ~ gruppe,  type='kaplan-meier',
            conf.type="plain", data=chemo))
tab
## Call: survfit(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     type = "kaplan-meier", conf.type = "plain")
##
##                 gruppe=1
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    93     15       1    0.933  0.0644        0.807        1.000
##   105     14       1    0.867  0.0878        0.695        1.000
##   108     13       1    0.800  0.1033        0.598        1.000
##   135     12       1    0.733  0.1142        0.510        0.957
##   242      8       1    0.642  0.1317        0.384        0.900
##   263      7       1    0.550  0.1412        0.273        0.827
##   518      4       1    0.413  0.1594        0.100        0.725
##   582      2       1    0.206  0.1662        0.000        0.532
##   595      1       1    0.000     NaN          NaN          NaN
##
##                 gruppe=2
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    30     14       1    0.929  0.0688       0.7937        1.000
##    55     13       1    0.857  0.0935       0.6738        1.000
##    89     10       1    0.771  0.1170       0.5420        1.000
##    90      9       1    0.686  0.1317       0.4275        0.944
##   101      8       1    0.600  0.1404       0.3248        0.875
```

```
##   148      7       1    0.514  0.1442       0.2317        0.797
##   155      6       1    0.429  0.1434       0.1476        0.710
##   233      4       1    0.321  0.1420       0.0431        0.600
##   283      2       1    0.161  0.1340       0.0000        0.423
```

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)

fit <- survfit(Surv(zeit, status) ~ gruppe,  type='kaplan-meier',
                    conf.type="plain", data=chemo)
par(mfcol=c(1,2),lwd=2, font.axis=2, bty="n", ps=14)
plot(fit[1], bg=0, xlim=c(0,600), ylim=c(0,1), las=1,
                xlab="Zeit in Tagen", ylab="S(t)", main=expression(C[1]))
abline(h=0.5, lty=3)
plot(fit[2], xlim=c(0,600), ylim=c(0,1), las=1,
                xlab="Zeit in Tagen", ylab="S(t)", main=expression(C[2]))
abline(h=0.5, lty=3)
```

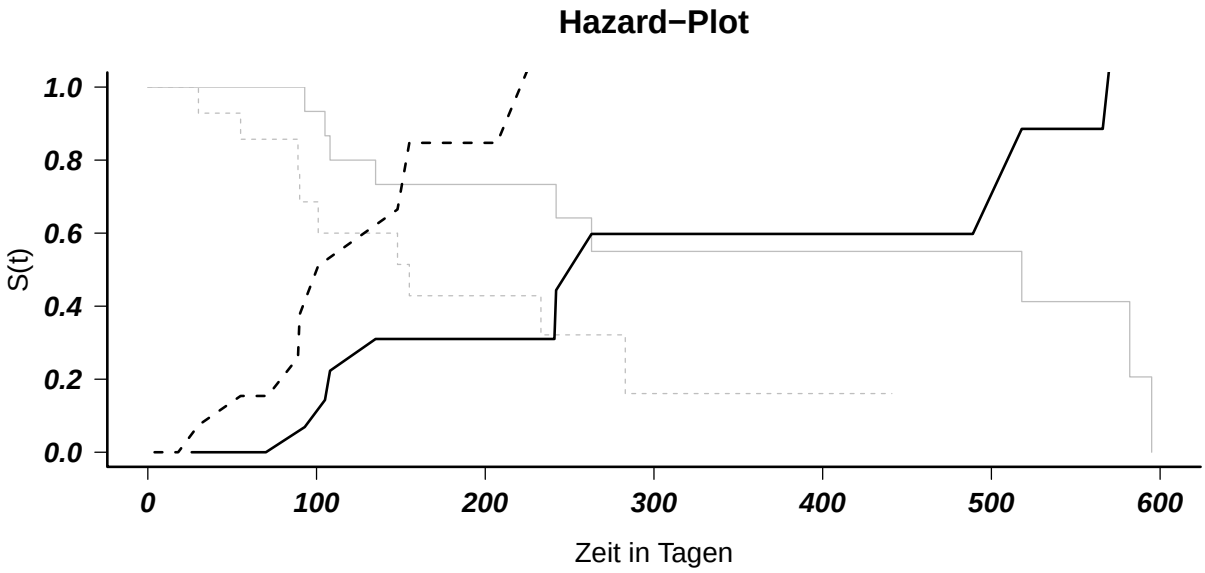


Mean und Median Survival:

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)

fit <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
print(fit,  print.rmean=TRUE)
## Call: survfit(formula = Surv(zeit, status) ~ gruppe, data = chemo)
##
##           n events *rmean *se(rmean) median 0.95LCL 0.95UCL
## gruppe=1 20      9    361       49.1    518     242      NA
## gruppe=2 18      9    210       48.9    155      90      NA
##     * restricted mean with upper limit =  518
```

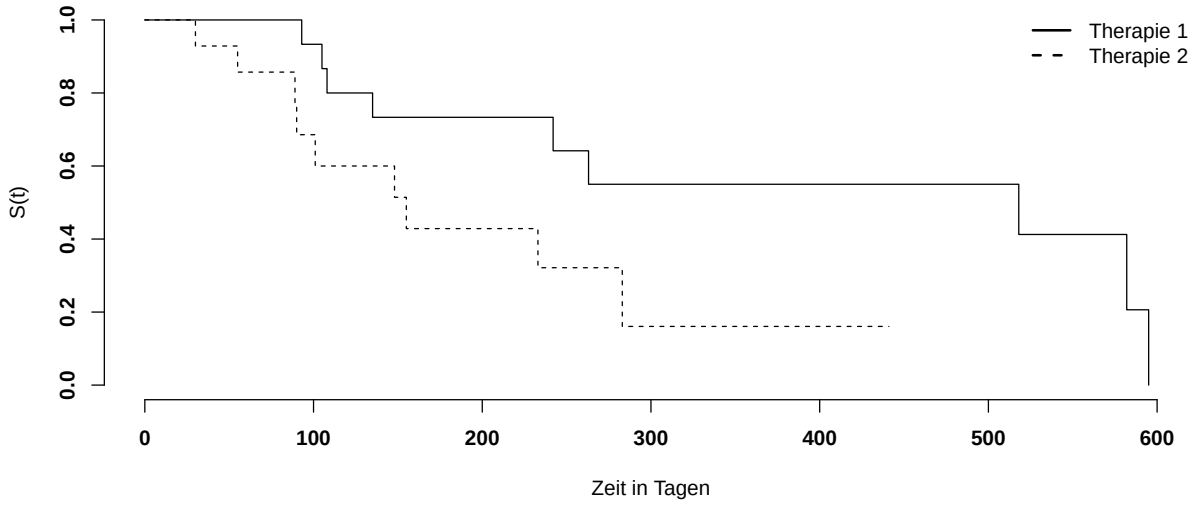
Plot kumulierte Hazards:

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)

par(mfrow=c(1,1), lwd=2, font.axis=4, bty="l", ps=16)
fit <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
H   <- -log(fit$surv)
plot(fit, conf.int=FALSE, col="grey", xlim=c(0,600), ylim=c(0,1), las=1,
      main="Hazard-Plot", lty=c(1,2), xlab="Zeit in Tagen", ylab="S(t)")
lines(chemo$zeit[chemo$gruppe==1], H[1:20], lty=1)
lines(chemo$zeit[chemo$gruppe==2], H[21:38], lty=2)
```



8.7.2 Der Logrank-Test

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)

fit <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)

par(mfcol=c(1,1), lwd=2, font.axis=2, bty="n", ps=12)
plot(fit, col=c(1,1), xlim=c(0,600), ylim=c(0,1),
           lty=c(1,2), xlab="Zeit in Tagen", ylab="S(t)")
legend("topright", c("Therapie 1","Therapie 2"),
           bty = "n", lty=c(1,2), col=c(1,1), cex=1)
```



Logrank-Test:

```
survdiff(Surv(zeit, status) ~ gruppe, data=chemo, rho=0)
## Call:
## survdiff(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     rho = 0)
##
##            N Observed Expected (O-E)^2/E (O-E)^2/V
## gruppe=1 20        9    12.73      1.09      4.12
## gruppe=2 18        9     5.27      2.64      4.12
##
##  Chisq= 4.1  on 1 degrees of freedom, p= 0.04
```

Wilcoxon-Test

```
survdiff(Surv(zeit, status) ~ gruppe, data=chemo, rho=1)
## Call:
## survdiff(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     rho = 1)
##
##            N Observed Expected (O-E)^2/E (O-E)^2/V
## gruppe=1 20     5.01     7.98      1.10      4.36
## gruppe=2 18     7.00     4.03      2.18      4.36
##
##  Chisq= 4.4  on 1 degrees of freedom, p= 0.04
```

8.7.3 Parametrische Regressionsmodelle für Überlebenszeiten

Beispiel Überleben nach Chemotherapie:

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)
chemo[1:10,]
```

X	gruppe	zeit	status
1	1	26	0
2	1	50	0
3	1	51	0
4	1	57	0
5	1	70	0
6	1	93	1
7	1	105	1
8	1	108	1
9	1	135	1
10	1	193	0

```
nc <- length(zeit)
```

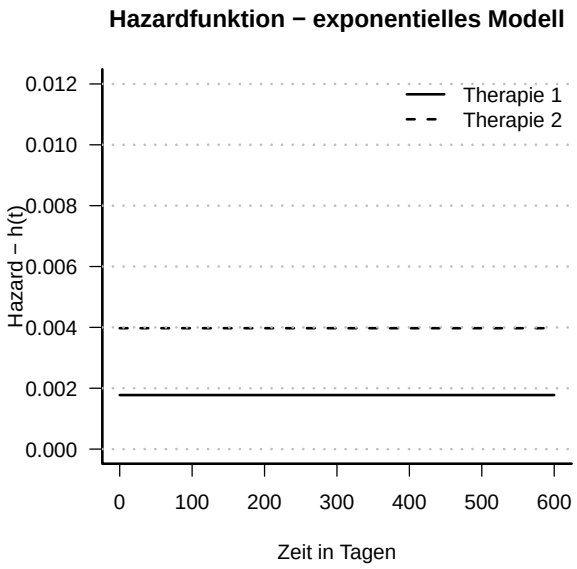
8.7.3.1 Exponentielles Modell

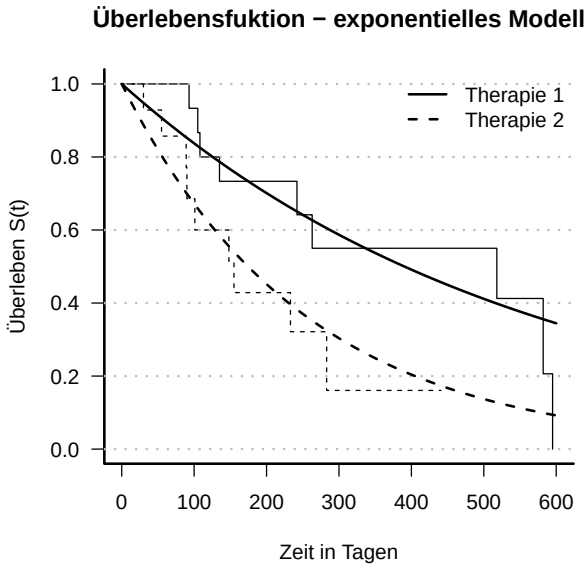
Beispiel Chremotherapie mit Funktion survreg() in library(survival):

```
library(survival)

chemo$gruppe <- as.factor(chemo$gruppe)
fit1 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "exponential")
print(summary(fit1))
##
## Call:
## survreg(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     dist = "exponential")
##               Value Std. Error     z       p
## (Intercept)  6.333      0.333 19.00 <2e-16
## gruppe2     -0.805      0.471 -1.71  0.088
##
## Scale fixed at 1
##
## Exponential distribution
## Loglik(model)= -124.8   Loglik(intercept only)= -126.2
##  Chisq= 2.84 on 1 degrees of freedom, p= 0.092
## Number of Newton-Raphson Iterations: 5
## n= 38
alpha.0 <- fit1$coefficients[1]                    # Koeffizienten
alpha.1 <- fit1$coefficients[2]
lp.1 <- alpha.0 + alpha.1 * 0                      # linearer Schätzer
```

```r
lp.2 <- alpha.0 + alpha.1 * 1

erw.1    <- exp(lp.1); erw.1                          # Erwartungswerte
## (Intercept)
##    563.1111
erw.2    <- exp(lp.2); erw.2
## (Intercept)
##    251.6667
```

```r
t  <- seq(0,600,by=10)
hazard.1  <- exp(-lp.1);  hazard.2  <- exp(-lp.2)  # Hazardfunktion

par(mfcol=c(1,2), lwd=2, font.axis=1.5, bty="n", ps=12, bty="l")
plot(t, rep(hazard.1, length(t)), type="l",
     main="Hazardfunktion - exponentielles Modell",
     xlim=c(0,600), ylim=c(0,0.012), las=1, lty=1,
     xlab="Zeit in Tagen", ylab="Hazard - h(t)")
lines(t, rep(hazard.2, length(t)), lty=2)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
abline(h=seq(0, 0.012, 0.002), col="grey", lty=3)

s.1 <- exp(-hazard.1 * t)                           # Überlebensfunktion
s.2 <- exp(-hazard.2 * t)

fitKM <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
plot(fitKM, col=c(1,1), xlim=c(0,600), ylim=c(0,1), las=1,
     main="Überlebensfuktion - exponentielles Modell",
     lty=c(1,2), xlab="Zeit in Tagen", ylab="Überleben S(t)")
abline(h=seq(0, 1, 0.2), col="grey", lty=3)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
lines(t, s.1, lty=1)
lines(t, s.2, lty=2)
```



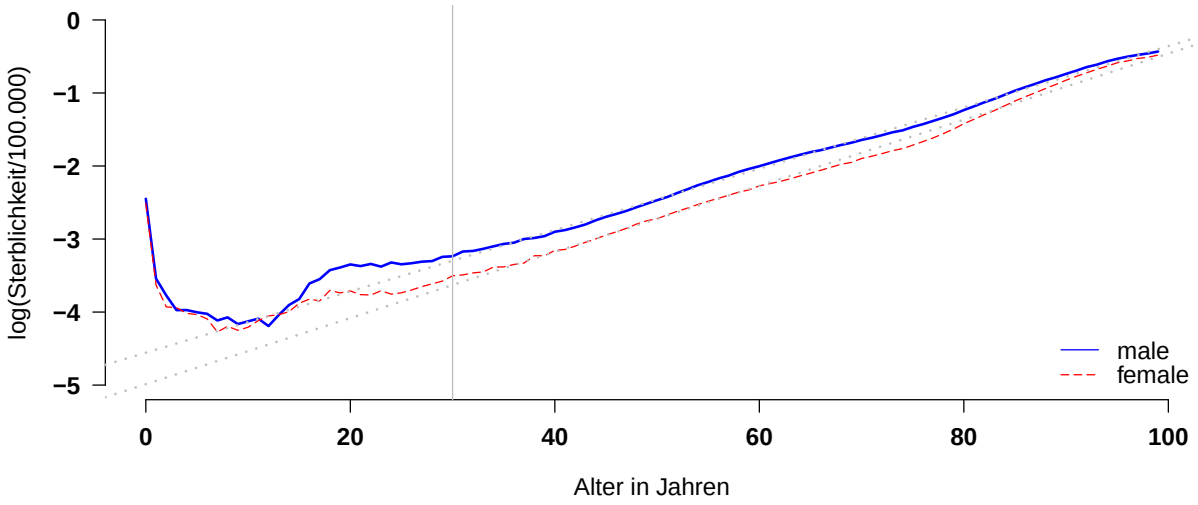


8.7.3.2 Gompertz-Regressionsmodell

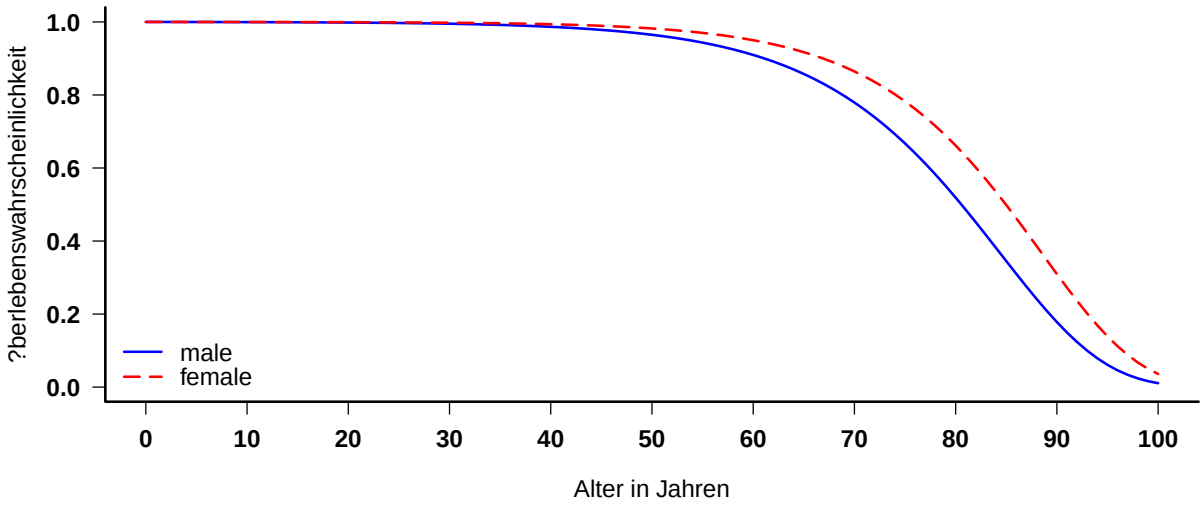
Beispiel Sterbetafeldaten 2015-2017:

Funktion read_excel() in library(readxl)

```
library(readxl)

dtab <- as.data.frame(read_excel("sterbetafel.xlsx"))
males   <- dtab[dtab$gender == 1,]
females <- dtab[dtab$gender == 2,]

par(mfcol=c(1,1), font.axis=2, bty="n", ps=14)
plot(males$age, log10(males$dprob), type="l", las=1, col="blue",
     ylab="log(Sterblichkeit/100.000)", xlab="Alter in Jahren",
     ylim=c(-5, 0), lwd=2, lty=1)
lines(females$age, log10(females$dprob), col="red", lty=5)
legend("bottomright", c("male","female"), bty="n",
       lty=c(1,5), col=c("blue","red"))

abline(v=30, col="grey")
mage30   <- males$age[males$age >= 30];         # Modell (>=30 Jahre)
mdpr30   <- males$dprob[males$age >= 30]
fage30   <- females$age[females$age >= 30];
fdpr30   <- females$dprob[females$age >= 30]
abline(lm(log10(mdpr30) ~ mage30), col="grey", lty=3, cex=0.3, lwd=2)
abline(lm(log10(fdpr30) ~ fage30), col="grey", lty=3, cex=0.3, lwd=2)
```




```
mmodel   <- lm(log(mdpr30) ~ mage30);
fmodel   <- lm(log(fdpr30) ~ fage30)
mlambda  <- exp(mmodel$coeff[1]);
flambda  <- exp(fmodel$coeff[1])
malpha   <- mmodel$coeff[2];
falpha   <- fmodel$coeff[2]
mmht     <- mlambda * exp(malpha*mage30)             # Hazard males
fmht     <- flambda * exp(falpha*fage30)             # Hazard females

par(lwd=2, font.axis=2, bty="l", ps=14, lab=c(10,5,2), las=1)
t   <- seq(0, 100, by=1)
mS  <- exp(mlambda/malpha *(1-exp(malpha*t)))
fS  <- exp(flambda/falpha *(1-exp(falpha*t)))
plot(t, mS, type="l", ylab="?berlebenswahrscheinlichkeit", lty=1,
     xlab="Alter in Jahren", xlim=c(0,100), ylim=c(0,1), col="blue")
lines(t, fS, col="Red", lty=5)
legend("bottomleft", c("male","female"), bty="n", lty=c(1,5), col=c("blue","red"))
```

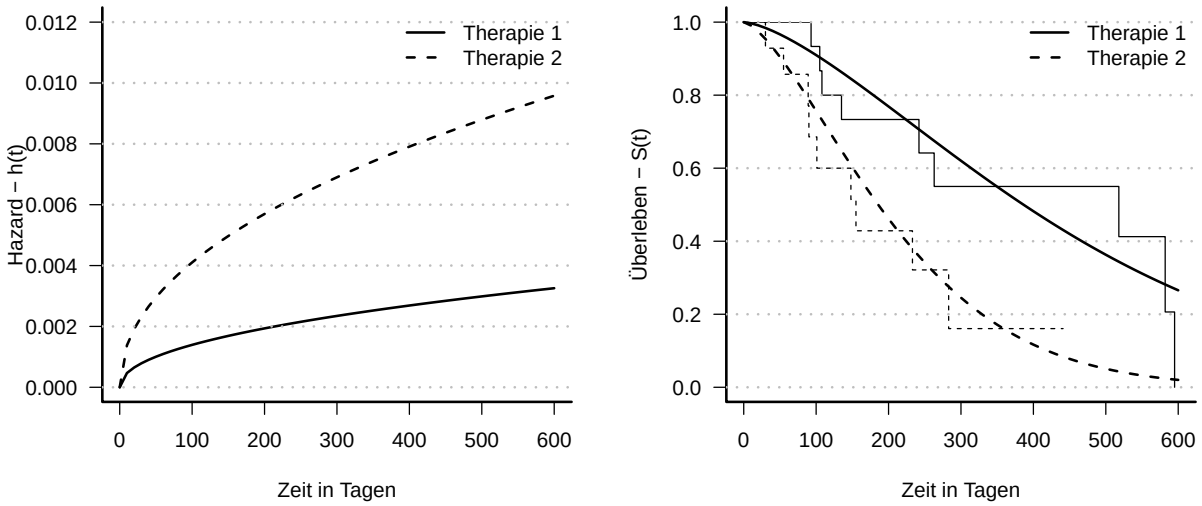


8.7.3.3 Weibull Regressionsmodell

Beispiel Chremotherapie mit Funktion survreg() in library(survival):

```
chemo <- read.csv2("chemotherapie.csv")
attach(chemo)

chemo$gruppe <- as.factor(chemo$gruppe)
fit2 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "weibull")
print(summary(fit2))
##
## Call:
## survreg(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     dist = "weibull")
##               Value Std. Error     z      p
## (Intercept)  6.206      0.229 27.12 <2e-16
## gruppe2     -0.733      0.321 -2.28  0.023
## Log(scale)  -0.388      0.181 -2.14  0.032
##
## Scale= 0.679
##
## Weibull distribution
## Loglik(model)= -122.8   Loglik(intercept only)= -125.1
##  Chisq= 4.61 on 1 degrees of freedom, p= 0.032
## Number of Newton-Raphson Iterations: 6
## n= 38
alpha.0 <- fit2$coefficients[1]                    # Koeffizienten
alpha.1 <- fit2$coefficients[2]
scale   <- fit2$scale; shape   <- 1/scale
lp.1 <- alpha.0 + alpha.1 * 0                      # linearer Schätzer
lp.2 <- alpha.0 + alpha.1 * 1
```

```r
erw.1    <- exp(lp.1) * gamma(1 + scale); erw.1      # Erwartungswerte
## (Intercept)
##    448.3189
erw.2    <- exp(lp.2) * gamma(1 + scale); erw.2
## (Intercept)
##    215.4708

t  <- seq(0,600,by=10)
hazard.1  <- shape * exp(-lp.1)^shape * t^(shape-1)  # Hazardfunktion
hazard.2  <- shape * exp(-lp.2)^shape * t^(shape-1)

par(mfcol=c(1,2), lwd=2, font.axis=1.5, bty="n", ps=12, bty="l")
plot(t, hazard.1, type="l",
     xlim=c(0,600), ylim=c(0,0.012), las=1, lty=1,
     xlab="Zeit in Tagen", ylab="Hazard - h(t)")
lines(t, hazard.2, lty=2)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
abline(h=seq(0, 0.012, 0.002), col="grey", lty=3)

s.1 <- exp(-(t*exp(-lp.1))^shape)                    # Überlebensfunktion
s.2 <- exp(-(t*exp(-lp.2))^shape)

fitKM <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
plot(fitKM, col=c(1,1), xlim=c(0,600), ylim=c(0,1), las=1,
     lty=c(1,2), xlab="Zeit in Tagen", ylab="Überleben - S(t)")
abline(h=seq(0, 1, 0.2), col="grey", lty=3)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
lines(t, s.1, lty=1)
lines(t, s.2, lty=2)
```

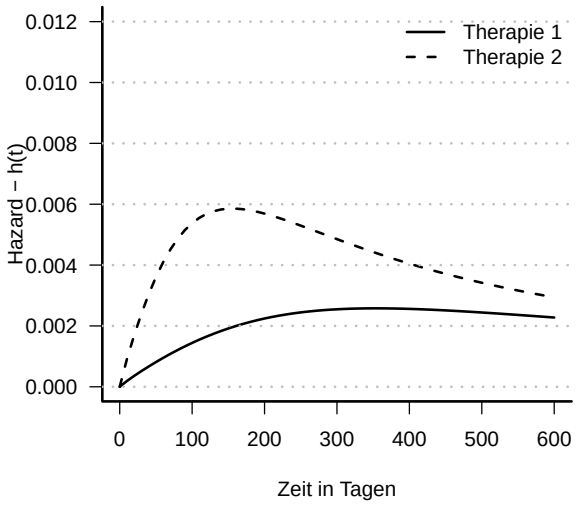


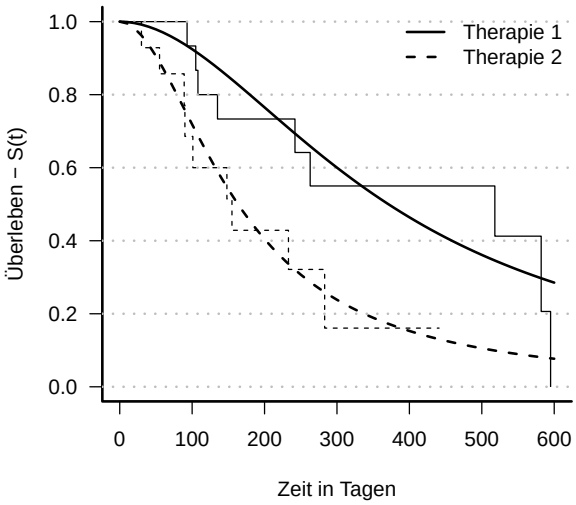
8.7.3.4 Loglogistisches Regressionsmodell

Beispiel Chremotherapie mit Funktion survreg() in library(survival):

```
chemo$gruppe <- as.factor(chemo$gruppe)
attach(chemo)

fit3 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "loglogistic")

print(summary(fit3))
##
## Call:
## survreg(formula = Surv(zeit, status) ~ gruppe, data = chemo,
##     dist = "loglogistic")
##               Value Std. Error     z       p
## (Intercept)  5.916      0.258 22.90 < 2e-16
## gruppe2     -0.820      0.364 -2.25 0.02438
## Log(scale)  -0.648      0.184 -3.52 0.00044
##
## Scale= 0.523
##
## Log logistic distribution
## Loglik(model)= -122.4   Loglik(intercept only)= -124.7
##  Chisq= 4.66 on 1 degrees of freedom, p= 0.031
## Number of Newton-Raphson Iterations: 5
## n= 38
alpha.0 <- fit3$coefficients[1]                    # Koeffizienten
alpha.1 <- fit3$coefficients[2]
scale   <- fit3$scale; shape <- 1/scale
lp.1 <- alpha.0 + alpha.1 * 0                      # linearer Schätzer
lp.2 <- alpha.0 + alpha.1 * 1
erw.1 <- exp(lp.1); erw.1                          # Erwartungswerte
## (Intercept)
##    371.0444
erw.2 <- exp(lp.2); erw.2
## (Intercept)
##    163.3414

t  <- seq(0,600,by=10)
                                                   # Hazardfunktionen
hazard.1 <- (shape * exp(-lp.1)^shape * t^(shape-1))/(1 + (exp(-lp.1)*t)^shape)
hazard.2 <- (shape * exp(-lp.2)^shape * t^(shape-1))/(1 + (exp(-lp.2)*t)^shape)

par(mfcol=c(1,2), lwd=2, font.axis=1.5, bty="n", ps=12, bty="l")
plot(t, hazard.1, type="l",
     xlim=c(0,600), ylim=c(0,0.012), las=1, lty=1,
     xlab="Zeit in Tagen", ylab="Hazard - h(t)")
lines(t, hazard.2, lty=2)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
abline(h=seq(0, 0.012, 0.002), col="grey", lty=3)

s.1 <- 1 / (1 + (exp(-lp.1)*t)^shape)              # Überlebensfunktion
```

```
s.2 <- 1 / (1 + (exp(-lp.2)*t)^shape)

fitKM <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
plot(fitKM, col=c(1,1), xlim=c(0,600), ylim=c(0,1), las=1,
     lty=c(1,2), xlab="Zeit in Tagen", ylab="Überleben - S(t)")
abline(h=seq(0, 1, 0.2), col="grey", lty=3)
legend("topright", c("Therapie 1","Therapie 2"),
       bty = "n", lty=c(1,2), col=c(1,1), cex=1)
lines(t, s.1, lty=1)
lines(t, s.2, lty=2)
```





8.7.3.5 Modellwahl und Güte der Anpassung

```
isy <- function(x) {                                  # check for infnity
  for (i in 1:length(x)) if (is.infinite(x[i])) x[i] <- NA
  return(x)        }
```

funktion survreg() in library(survival)

```
chemo$gruppe <- as.factor(chemo$gruppe)
attach(chemo)

library(survival)
fitKM <- survfit(Surv(zeit, status) ~ gruppe, data=chemo)
n1 <- fitKM$n[1];        n2 <- fitKM$n[2]
t1 <- fitKM$time[1:n1];  t2 <- fitKM$time[(n1+1):(n1+n2)]
s1 <- fitKM$surv[1:n1];  s2 <- fitKM$surv[(n1+1):(n1+n2)]

cat("\n","---> exponential model","\n")
##
```

```
##  ---> exponential model
fit1 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "exponential")

cat("\n","---> Weibull model","\n")
##
##  ---> Weibull model
fit2 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "weibull")

cat("\n","---> loglogistic model","\n")
##
##  ---> loglogistic model
fit3 <- survreg(Surv(zeit, status) ~ gruppe,
                data = chemo, dist = "loglogistic")


par(mfcol=c(1,3), lwd=2, font.axis=1.8, bty="n", ps=18, bty="l")


l.1 <- coef(fit1)[1] + coef(fit1)[2]*0             # Exponential-Modell
l.2 <- coef(fit1)[1] + coef(fit1)[2]*1
h.1 <- exp(-l.1)                                   # Hazard
h.2 <- exp(-l.2)
s.1 <- exp(-exp(-l.1) * t1)                        # Überlebensfunktion
s.2 <- exp(-exp(-l.2) * t2)

y.1 <- -log(s1); y.2 <- -log(s2)
x.1 <- t1;       x.2 <- t2
plot(x.1, y.1,  type="p", pch=1, cex=1.5, las=1,
     xlim=c(0, 600), ylim=c(0,1),
     main ="exponentielles Modell",
     ylab="-log(S(t))",
     xlab="Überlebenszeit t")
abline(a=0, b=h.1, lty=1)
points(x.2, y.2, pch=2, cex=1.5)
abline(a=0, b=h.2, lty=2)
legend("bottomright", c("Therapie 1","Therapie 2"),
       bty = "n", pch=c(1,2), col=c(1,1), cex=1)

l.1 <- coef(fit2)[1] + coef(fit2)[2]*0             # Weibull model
l.2 <- coef(fit2)[1] + coef(fit2)[2]*1;  shape <- 1/fit2$scale
h.1 <- exp(-l.1)^shape                             # Hazard
h.2 <- exp(-l.2)^shape
s.1 <- exp(-(t1*exp(-l.1))^shape)                  # Überlebensfunktion
s.2 <- exp(-(t2*exp(-l.2))^shape)

y.1 <- isy(log(-log(s1)));      y.2 <- isy(log(-log(s2)))
x.1 <- log(t1);                 x.2 <- log(t2)
plot(x.1, y.1,  type="p", pch=1, cex=1.5,las=1,
     xlim=c(3, 6.5), ylim=c(-4, 1),
     main ="Weibull Modell",
     ylab="log(-log(S(t)))",
```

```
     xlab="Überlebenszeit log(t)")
abline(a=log(h.1), b=shape, lty=1)
points(x.2, y.2, pch=2, cex=1.5)
abline(a=log(h.2), b=shape, lty=2)
legend("bottomright", c("Therapie 1","Therapie 2"),
       bty = "n", pch=c(1,2), col=c(1,1), cex=1)

l.1 <- coef(fit3)[1] + coef(fit3)[2]*0            # loglogistic model
l.2 <- coef(fit3)[1] + coef(fit3)[2]*1;   shape <- 1/fit3$scale
h.1 <- exp(-l.1)^shape                            # Hazard
h.2 <- exp(-l.2)^shape
s.1 <- 1 / (1 + (exp(-l.1)*t1)^shape)             # Überlebensfunktion
s.2 <- 1 / (1 + (exp(-l.2)*t2)^shape)

y.1 <- isy(log((1-s1)/s1));     y.2 <- isy(log((1-s2)/s2))
x.1 <- log(t1);                 x.2 <- log(t2)
plot(x.1, y.1,  type="p", pch=1, cex=1.5,las=1,
     xlim=c(3, 6.5), ylim=c(-4, 1),
     main ="loglogistisches Modell",
     ylab="log((1-S(t))/S(t))",
     xlab="Überlebenszeit log(t)")
abline(a=log(h.1), b=shape, lty=1)
points(x.2, y.2, pch=2, cex=1.5)
abline(a=log(h.2), b=shape, lty=2)
legend("bottomright", c("Therapie 1","Therapie 2"),
       bty = "n", pch=c(1,2), col=c(1,1), cex=1)
```

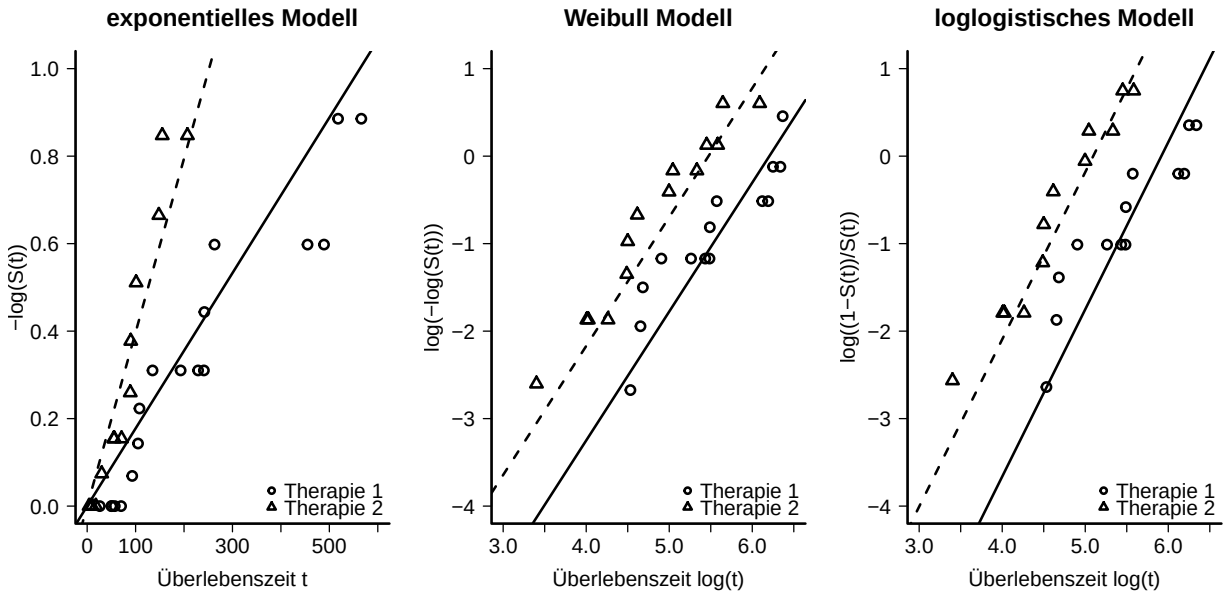


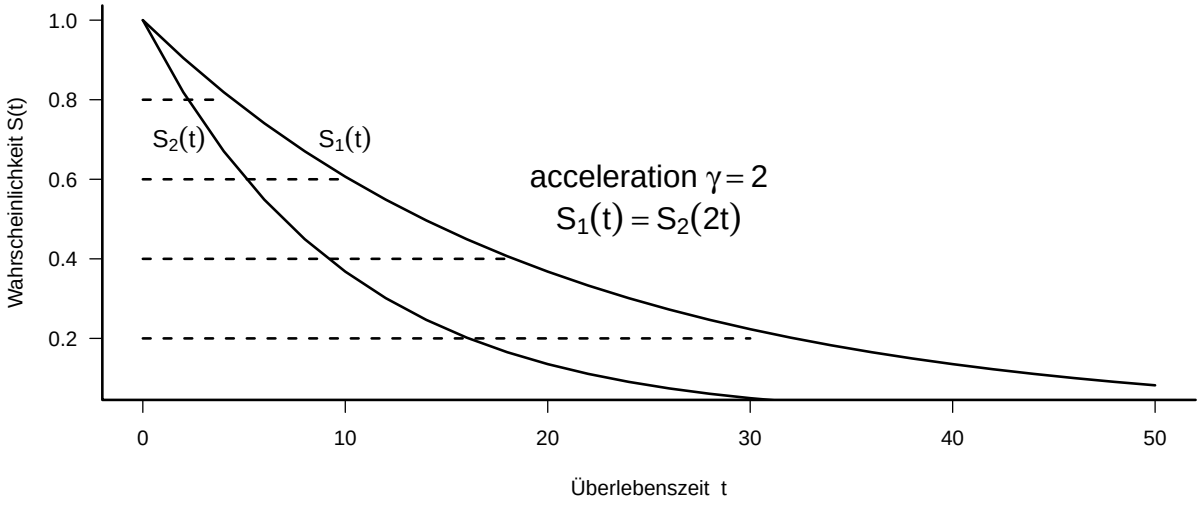
Tabelle:

```
chemo$gruppe <- as.factor(chemo$gruppe)
attach(chemo)
nc <- length(chemo$zeit)
                                                # exponentielles Modell
L0  <- fit1$loglik[1]; L <- fit1$loglik[2]                  # Likelihood
G1  <- 2*(L - L0); p1 <- pchisq(G1, 1, lower.tail=FALSE)    # G-Statistik
RC1 <-  1-(exp(L0)/exp(L))^(2/nc)                           # Cox-Snell
RN1 <- RC1 / (1-(exp(L0)^(2/nc)))                           # Nagelkerke
                                                # Weibull Modell
L0  <- fit2$loglik[1]; L <- fit2$loglik[2]                  # Likelihood
G2  <- 2*(L - L0); p2 <- pchisq(G2, 1, lower.tail=FALSE)    # G-Statistik
RC2 <-  1-(exp(L0)/exp(L))^(2/nc);                          # Cox-Snell
RN2 <- RC2 / (1-(exp(L0)^(2/nc)))                           # Nagelkerke
                                                # loglogistisches Modell
L0  <- fit3$loglik[1]; L <- fit3$loglik[2]                  # Likelihood
G3  <- 2*(L - L0); p3 <- pchisq(G3, 1, lower.tail=FALSE)    # G-Statistik
RC3 <-  1-(exp(L0)/exp(L))^(2/nc)                           # Cox-Snell
RN3 <- RC3 / (1-(exp(L0)^(2/nc)))                           # Nagelkerke

tab  <- matrix(data = NA, nrow = 3, ncol = 3, byrow = FALSE,
               dimnames = NULL)
tab[1,1] <- round(G1, 4); tab[1,2] <-  round(p1, 4)
tab[1,3] <-  round(RN1, 4)
tab[2,1] <- round(G2, 4); tab[2,2] <-  round(p2, 4)
tab[2,3] <-  round(RN2, 4)
tab[3,1] <- round(G3, 4); tab[3,2] <-  round(p3, 4)
tab[3,3] <-  round(RN3, 4)
rownames(tab) <- c("exponent. Modell","Weibull Modell","loglogist. Modell")
colnames(tab) <- c("G","(p)","R")
as.data.frame(tab)
```

	G	(p)	R
exponent. Modell	2.8432	0.0918	0.0722
Weibull Modell	4.6123	0.0317	0.1145
loglogist. Modell	4.6568	0.0309	0.1155

8.7.3.6 AFT-Modelle

```
t      <- seq(0, 50, 2)
hazard <- 0.05
par(mfcol=c(1,1), lwd=2, font.axis=1.5, bty="n", ps=12, bty="l")
surv1   <- exp(-hazard*t)
plot(t, surv1, type="l", las=1,
     xlab="Überlebenszeit  t", ylab="Wahrscheinlichkeit S(t)")
surv2   <- exp(-hazard*2*t)
lines(t,surv2)
text(25, 0.6, expression(paste(plain("acceleration "), gamma == 2)), cex=1.5)
text(25, 0.5, expression(S[1](t) == S[2](2 * t)), cex=1.5)
text(10, 0.7, expression(S[1](t)), cex=1.2)
text(1.8, 0.7, expression(S[2](t)), cex=1.2)
lines(t[1:3], rep(0.8, 3), lty=2)
lines(t[1:6], rep(0.6, 6), lty=2)
lines(t[1:10], rep(0.4, 10), lty=2)
lines(t[1:16], rep(0.2, 16), lty=2)
```



8.7.4 Das Proportional-Hazards Modell von Cox

8.7.4.1 Parameterschätzung

Beispiel Ovarialkarzinom mit Funktion coxph() in library(survival):

```
library(survival)
data(ovarian)
attach(ovarian)
ovarian[1:10,]
```

futime	fustat	age	resid.ds	rx	ecog.ps
59	1	72.3315	2	1	1
115	1	74.4932	2	1	1
156	1	66.4658	2	1	2
421	0	53.3644	2	2	1
431	1	50.3397	2	1	1
448	0	56.4301	1	1	2
464	1	56.9370	2	2	2
475	1	59.8548	2	2	2
477	0	64.1753	2	1	1
563	1	55.1781	1	2	2

```
summary(survfit(Surv(futime, fustat)~1 , data=ovarian))
## Call: survfit(formula = Surv(futime, fustat) ~ 1, data = ovarian)
##
##  time n.risk n.event survival std.err lower 95% CI upper 95% CI
##    59     26       1    0.962  0.0377        0.890        1.000
##   115     25       1    0.923  0.0523        0.826        1.000
##   156     24       1    0.885  0.0627        0.770        1.000
##   268     23       1    0.846  0.0708        0.718        0.997
##   329     22       1    0.808  0.0773        0.670        0.974
##   353     21       1    0.769  0.0826        0.623        0.949
##   365     20       1    0.731  0.0870        0.579        0.923
##   431     17       1    0.688  0.0919        0.529        0.894
##   464     15       1    0.642  0.0965        0.478        0.862
##   475     14       1    0.596  0.0999        0.429        0.828
##   563     12       1    0.546  0.1032        0.377        0.791
##   638     11       1    0.497  0.1051        0.328        0.752
```

```
fit <- coxph(Surv(futime, fustat) ~ age + rx + resid.ds + ecog.ps,
                      data=ovarian)
summary(fit)
## Call:
## coxph(formula = Surv(futime, fustat) ~ age + rx + resid.ds +
##     ecog.ps, data = ovarian)
##
##   n= 26, number of events= 12
##
##               coef exp(coef) se(coef)      z Pr(>|z|)
## age       0.12481   1.13294  0.04689  2.662  0.00777 **
```

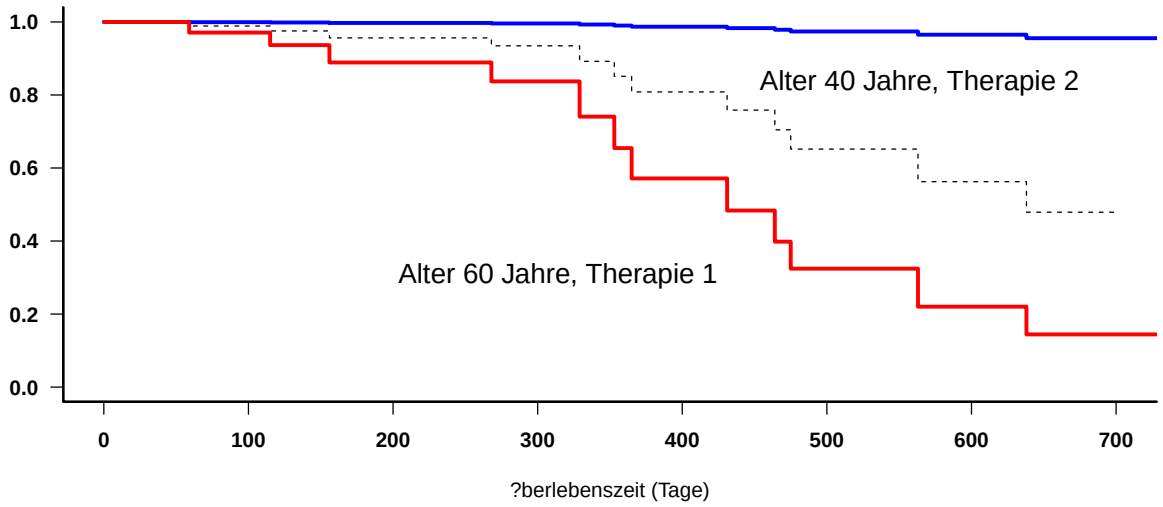
```
## rx       -0.91450   0.40072  0.65332 -1.400  0.16158
## resid.ds  0.82619   2.28459  0.78961  1.046  0.29541
## ecog.ps   0.33621   1.39964  0.64392  0.522  0.60158
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##          exp(coef) exp(-coef) lower .95 upper .95
## age         1.1329     0.8827    1.0335     1.242
## rx          0.4007     2.4955    0.1114     1.442
## resid.ds    2.2846     0.4377    0.4861    10.738
## ecog.ps     1.3996     0.7145    0.3962     4.945
##
## Concordance= 0.807  (se = 0.068 )
## Likelihood ratio test= 17.04  on 4 df,   p=0.002
## Wald test            = 14.25  on 4 df,   p=0.007
## Score (logrank) test = 20.81  on 4 df,   p=3e-04
```

8.7.4.2 Interpretation der Parameter

Modellrechnungen - Vorhersage:

Beispiel Ovarialkarzinom mit Funktion coxph() in library(survival):

```
library(survival)
data(ovarian)

fit <- coxph(Surv(futime,fustat) ~ age + rx, ovarian)
fit
## Call:
## coxph(formula = Surv(futime, fustat) ~ age + rx, data = ovarian)
##
##         coef exp(coef) se(coef)      z       p
## age  0.14733   1.15873  0.04615  3.193 0.00141
## rx  -0.80397   0.44755  0.63205 -1.272 0.20337
##
## Likelihood ratio test=15.89  on 2 df, p=0.0003551
## n= 26, number of events= 12

par(mfcol=c(1,1),lwd=2, font.axis=2, bty="l", ps=12)
plot( survfit(fit), conf.int=FALSE, lty=2, las=1,
        xlim=c(0,700), xlab="?berlebenszeit (Tage)")
lines( survfit(fit, newdata=data.frame(age=40, rx=2)),
       conf.int=FALSE, col="blue", lwd=3)
lines( survfit(fit, newdata=data.frame(age=60, rx=1)),
       conf.int=FALSE, col="red", lwd=3)
legend(420, 0.93, "Alter 40 Jahre, Therapie 2", bty="n", cex=1.3)
legend(170, 0.40, "Alter 60 Jahre, Therapie 1", bty="n", cex=1.3)
```



Nomogramm mit Funktion nomogram() in library(rms):

```
library(survival)
library(rms)
data(ovarian)

df <- ovarian
df$rx        <- factor(df$rx, levels=c(1,2), labels=c("group A","group B"))
names(df)[5] <- "treatment"
d <- datadist(df); options(datadist="d")

fit <- cph(Surv(futime,fustat) ~ age + treatment, surv=T, data = df)
surv   <- Survival(fit)
surv1  <- function(lp) surv(365, lp)                    # 1 Jahr Überleben
at.surv <- c(0.05, 0.25, 0.5, 0.75, 0.95, 0.98)

nom <- nomogram(fit, conf.int=F, fun=list(surv1),
                funlabel=c('1-year survival probability'), lp=F,
                fun.at=c(at.surv))

# plot(nom, xfrac=0.45)

print(nom)
## Points per unit of linear predictor: 16.9691
## Linear predictor units per point   : 0.05893064
##
##
##  age Points
##  35    0
##  40   13
##  45   25
##  50   38
##  55   50
```

```
##  60   62
##  65   75
##  70   88
##  75  100
##
##
##  treatment Points
##  group A   14
##  group B    0
##
##
##  Total Points 1-year survival probability
##           105                         0.05
##            92                         0.25
##            80                         0.50
##            65                         0.75
##            36                         0.95
##            20                         0.98
```

8.7.4.3 Modellbildung - Variablenauswahl

```
fitm <- coxph(Surv(futime, fustat) ~
                  age + rx + resid.ds + ecog.ps, ovarian)
fitm$loglik[1]; fitm$loglik[2]
## [1] -34.98494
## [1] -26.46329

fit1 <- update(fitm, . ~ . -ecog.ps)
gm   <- 2*(fitm$loglik[2]-fitm$loglik[1]); gm;
## [1] 17.04329
pchisq(gm, 4, lower.tail=FALSE)
## [1] 0.001895867
g1   <- 2*(fit1$loglik[2]-fit1$loglik[1]); g1;
## [1] 16.76757
pchisq(g1, 3, lower.tail=FALSE)
## [1] 0.0007889437
fit2 <- update(fit1, . ~ . - resid.ds)
g2   <- 2*(fit2$loglik[2]-fit2$loglik[1]); g2;
## [1] 15.88608
pchisq(g2, 2, lower.tail=FALSE)
## [1] 0.0003551247
fit3 <- update(fit2, . ~ . - age)
g3   <- 2*(fit3$loglik[2]-fit3$loglik[1]); g3;
## [1] 1.051453
pchisq(g3, 2, lower.tail=FALSE)
## [1] 0.5911257
```

Funktion stepAIC() in library(MASS):

```
library(MASS)
fit  <- coxph(Surv(futime, fustat) ~
                  age + rx + resid.ds + ecog.ps, data=ovarian)

summary(stepAIC(fit, upper = ~ age + rx + resid.ds + ecog.ps,trace=TRUE))
## Start:  AIC=60.93
## Surv(futime, fustat) ~ age + rx + resid.ds + ecog.ps
##
##             Df    AIC
## - ecog.ps    1 59.202
## - resid.ds   1 60.055
## - rx         1 60.868
## <none>         60.927
## - age        1 69.939
##
## Step:  AIC=59.2
## Surv(futime, fustat) ~ age + rx + resid.ds
##
##             Df    AIC
## - resid.ds   1 58.084
## - rx         1 58.942
## <none>         59.202
## - age        1 68.537
##
## Step:  AIC=58.08
## Surv(futime, fustat) ~ age + rx
##
##        Df    AIC
## - rx    1 57.676
## <none>    58.084
## - age   1 70.918
##
## Step:  AIC=57.68
## Surv(futime, fustat) ~ age
##
##        Df    AIC
## <none>    57.676
## - age   1 69.970
## Call:
## coxph(formula = Surv(futime, fustat) ~ age, data = ovarian)
##
##   n= 26, number of events= 12
##
##         coef exp(coef) se(coef)     z Pr(>|z|)
## age 0.16162   1.17541  0.04974 3.249  0.00116 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## age     1.175     0.8508     1.066     1.296
##
## Concordance= 0.784  (se = 0.083 )
```

```
## Likelihood ratio test= 14.29  on 1 df,   p=2e-04
## Wald test            = 10.56  on 1 df,   p=0.001
## Score (logrank) test = 12.26  on 1 df,   p=5e-04
```

Güte der Vorhersage-Konkordanzindex:

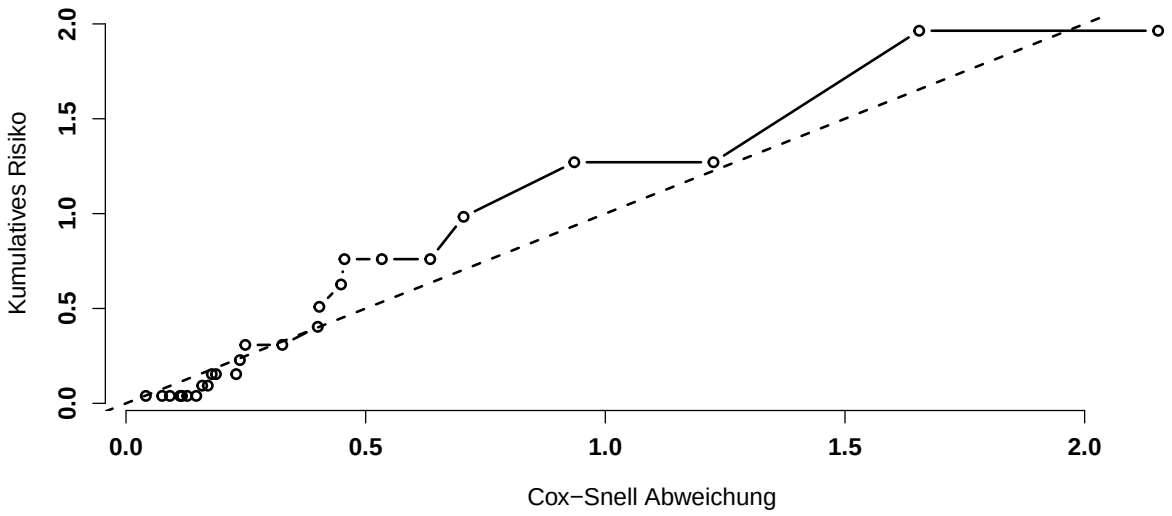
```
library(survival)
data(ovarian)

fit <- coxph(Surv(futime,fustat) ~ age + rx, ovarian)
summary(fit)
## Call:
## coxph(formula = Surv(futime, fustat) ~ age + rx, data = ovarian)
##
##   n= 26, number of events= 12
##
##         coef exp(coef) se(coef)      z Pr(>|z|)
## age  0.14733   1.15873  0.04615  3.193  0.00141 **
## rx  -0.80397   0.44755  0.63205 -1.272  0.20337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##     exp(coef) exp(-coef) lower .95 upper .95
## age    1.1587      0.863    1.0585     1.268
## rx     0.4475      2.234    0.1297     1.545
##
## Concordance= 0.798  (se = 0.076 )
## Likelihood ratio test= 15.89  on 2 df,   p=4e-04
## Wald test            = 13.47  on 2 df,   p=0.001
## Score (logrank) test = 18.56  on 2 df,   p=9e-05

concordance(fit)
## Call:
## concordance.coxph(object = fit)
##
## n= 26
## Concordance= 0.7982 se= 0.07633
## concordant discordant     tied.x     tied.y    tied.xy
##        174         44          0          0          0
```

8.7.4.4 Residuenanalyse - Güte der Modellanpassung

Cox-Snell Residuen:

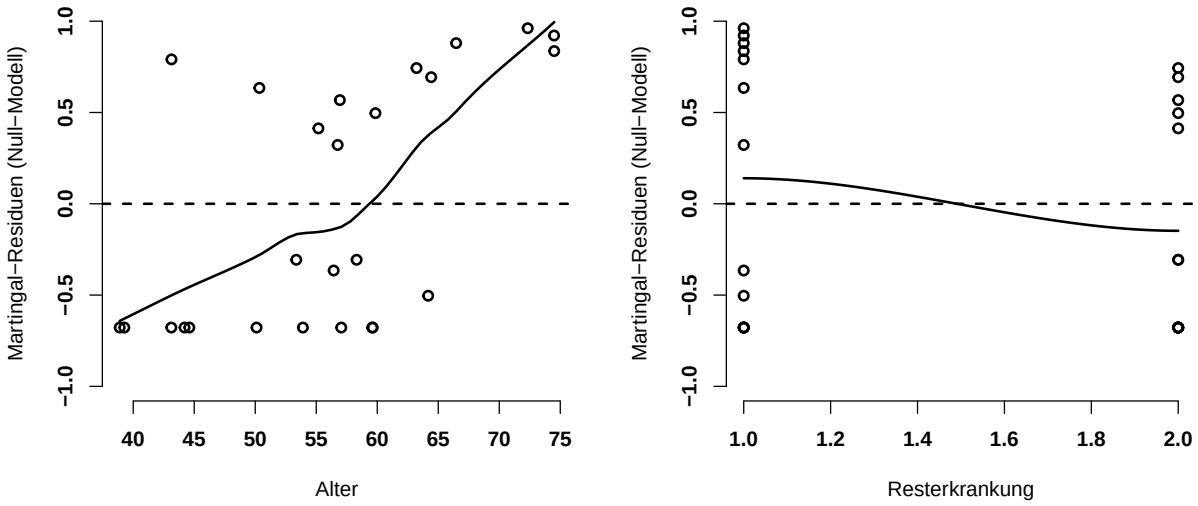
Beispiel Ovarialkarzinom mit Funktion coxph() in library(survival):

```
library(survival)
data(ovarian)

fit0 <- coxph(Surv(futime, fustat) ~ 1, ovarian)
fitm <- coxph(Surv(futime, fustat) ~ age + rx + resid.ds + ecog.ps, ovarian)
m.resid  <- resid(fitm)
cs.resid <- ovarian$fustat - m.resid
km.cs    <- survfit(Surv(cs.resid, ovarian$fustat)~1)
cs.times <- km.cs$time
cs.S     <- km.cs$surv
cs.exp   <- -log(cs.S)

par(mfcol=c(1,1),lwd=2, font.axis=2, bty="n", ps=14)
plot(cs.times, cs.exp, type="b", xlab="Cox-Snell Abweichung",
                ylab="Kumulatives Risiko")
abline(0, 1, lty=2)
```



Martingale Residuen:

```
library(survival)
data(ovarian)

par(mfcol=c(1,2),lwd=2, font.axis=2, bty="n", ps=12)
fit0 <- coxph(Surv(futime, fustat) ~ 1, ovarian)
scatter.smooth(ovarian$age, resid(fit0), xlab="Alter", ylim=c(-1,1),
ylab="Martingal-Residuen (Null-Modell)"); abline(h=0, lty=2)
scatter.smooth(ovarian$rx, resid(fit0), xlab="Resterkrankung", ylim=c(-1,1),
ylab="Martingal-Residuen (Null-Modell)"); abline(h=0, lty=2)
```



Schoenfeld Residuen:

```
fit.age <- coxph(Surv(futime, fustat)~age, data=ovarian)
detail <-  coxph.detail(fit.age)
time   <- detail$y[,2]
stat   <- detail$y[,3]
res    <- resid(fit.age, type="schoenfeld")

par(mfcol=c(1,1),lwd=2, font.axis=2, bty="n", ps=14)
scatter.smooth(time[stat==1], res, ylim=c(-20,10),
               xlab="Überlebenszeit", ylab="Schoenfeld Residuen zum Alter")
abline(h=0, lty=2)
```