AS17: Kapitel 6 - Schätzen

Jürgen Hedderich

2020-06-20

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_1.0.4.6    stringi_1.4.6   rmarkdown_2.3
##  [9] knitr_1.28      stringr_1.4.0   xfun_0.14       digest_0.6.25
## [13] rlang_0.4.6     evaluate_0.14
```

6.2 Zufallsstichproben und Zufallszahlen

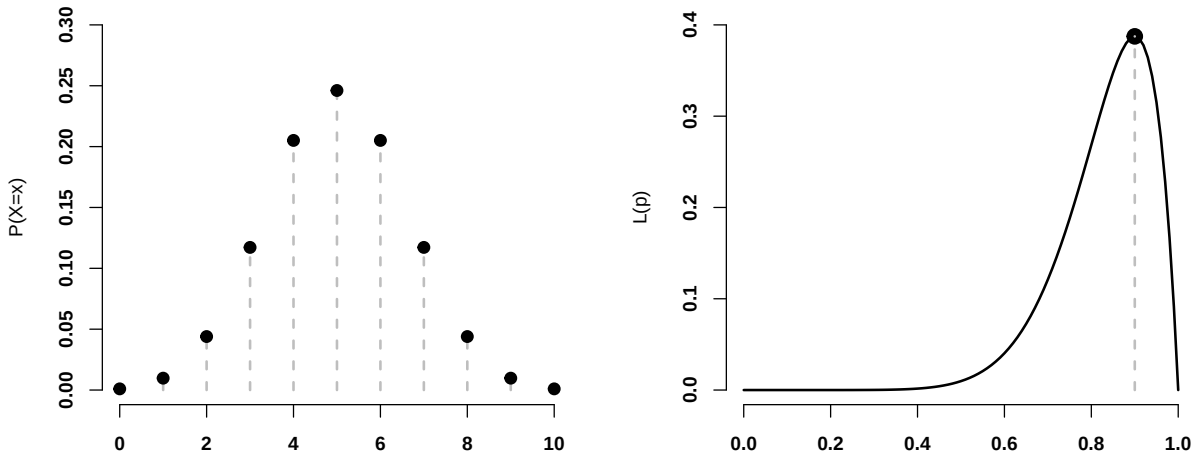
```
set.seed(123)

runif(5, min=0, max=1)                                  # gleichverteilt
## [1] 0.2875775 0.7883051 0.4089769 0.8830174 0.9404673

rnorm(5, mean=0, sd=1)                                  # standardnormalverteilt
## [1] -1.6895557  1.2394959 -0.1089660 -0.1172420  0.1830826

sample(1:80, 20, replace=FALSE)                         # Stichprobe ohne Zurücklegen
##  [1] 69 57  9 72 26  7 42 78 36 43 15 32 75 73 41 23 27 60 53 68

sample(1:80, 20, replace=TRUE)                          # Stichprobe ohne Zurücklegen
##  [1] 53 27 38 34 69 72 76 63 13 25 38 21 79 41 47 60 16  6 72 39
```

6.4 Schätzverfahren für Maßzahlen einer Verteilung

6.4.2 Schätzung nach der größten Ewratung

Beispiel Münzwurf:

```
n  <- 10; p  <- 1/2; x  <- 0:n
fx <- dbinom(x, n, p)

par(mfrow=c(1,2), lwd=2, font.axis=2, bty="n", ps=11)
plot(x, fx, type="h", ylim=c(0, 0.3), xlim=c(0,n),
                              ylab="P(X=x)", xlab=" ", lty=2, col="grey")
points(x, fx, pch=19, cex=1.1, col="black")

p  <- seq(0, 1, by=0.01)
Lx <- dbinom(9, n, p)
plot(p, Lx, type="l", ylim=c(0,0.4), xlim=c(0,1), ylab="L(p)", xlab=" ")
points(0.9, dbinom(9, n, 0.9), pch=19, cex=1.5, col="black")
lines(c(0.9,0.9), c(0,dbinom(9, n, 0.9)), lty=2, col="grey")
```



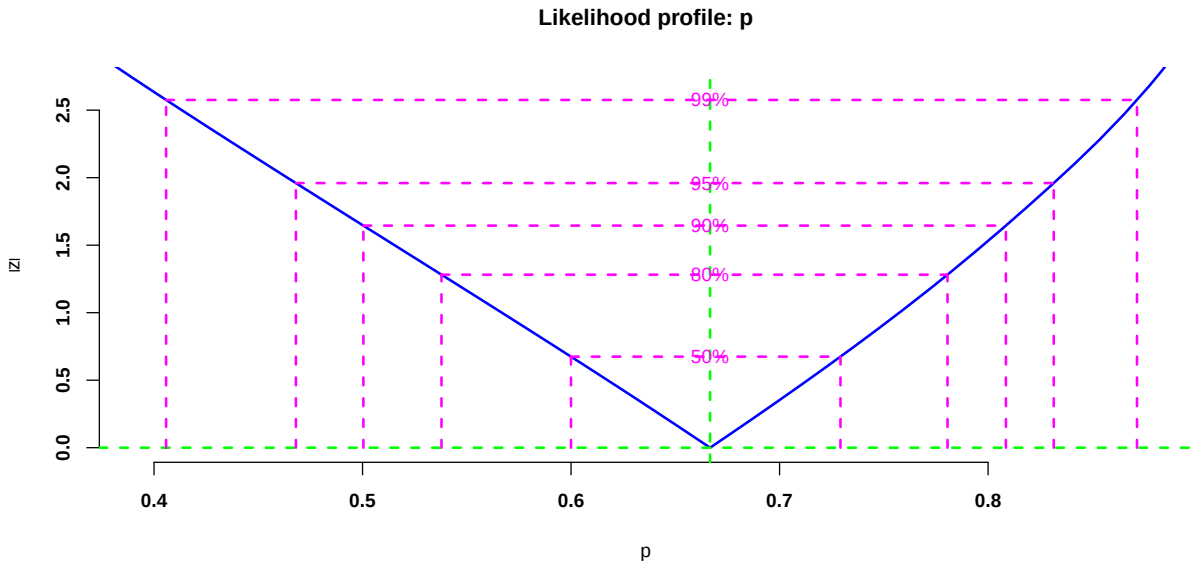
6.4.2.1 ML-Schätzer zur Binomialverteilung

Beispiel Münzwurf:

Funktion mle2() in library(bbmle)

```
library(bbmle)
x    <- 16                                              # Beobachtung: 16mal die Sechs
size <- 24                                              # Anzahl der Würfe (24)

                                                        # p=1/6 initial (ideal)
logL <- function(p = 1/6) -sum(stats::dbinom(x,  size, p, log = TRUE))

mle2(logL)
##
## Call:
## mle2(minuslogl = logL)
##
## Coefficients:
##         p
## 0.6666661
##
## Log-likelihood: -1.77

est  <- mle2(logL); summary(est)                        # Funktionen in library(bbmle)
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = logL)
##
## Coefficients:
##   Estimate Std. Error z value     Pr(z)
## p 0.666666   0.096225  6.9282 4.263e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 3.536147

par(mfrow=c(1,1), lwd=2, font.axis=2, bty="n", ps=11)  # Likelihood - Profile
profile(est); plot(profile(est))
## Likelihood profile:
##
## $p
##              z         p
## 1  -3.2312799 0.3425375
## 2  -2.6696081 0.3965590
## 3  -2.1308695 0.4505804
## 4  -1.6041055 0.5046018
## 5  -1.0798069 0.5586233
## 6  -0.5486077 0.6126447
## 7   0.0000000 0.6666661
## 8   0.5793918 0.7206876
## 9   1.2090029 0.7747090
## 10  1.9212278 0.8287304
```

```
## 11  2.7809137 0.8827519
## 12  3.9640668 0.9367733
```




```
confint(est)                                          # Konfidenzgrenzen
##     2.5 %    97.5 %
## 0.4680326 0.8313844
```

6.4.2.2 ML-Schätzer zur Negativen Binomilverteilung

Beispiel Karies:

Funktion mle2() in library(bbmle)

```
d3f <- 0:47
n   <- c(221, 32, 42, 27, 27, 13, 11, 9, 8, 14, 6, 5, 4, 7,
           6, 4, 4, 1, 1, 3, 3, 3, 3, 0, 1, 1, 0, 1, 1, 0, 0,
           1, 1, 0, 1, 1, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1)
N   <- sum(n)                                         # Momentenschätzung

m   <- sum(n*d3f)/N; m                                # Mittelwert
## [1] 3.989293

v   <- (sum(n*(d3f^2))-(sum(n*d3f))^2/N)/(N-1); v     # Varianz
## [1] 48.82607

prob <- m/v; prob                                     # p geschätzt
## [1] 0.08170417

size <- m^2/(v-m); size                               # k geschätzt
## [1] 0.3549422
```

```r
library(bbmle)
logL <- function(k=size, p=prob) -sum(stats::dnbinom(n, k, p, log=TRUE))
summary(mle2(logL))
## Maximum likelihood estimation
##
## Call:
## mle2(minuslogl = logL)
##
## Coefficients:
##    Estimate Std. Error z value     Pr(z)
## k 0.2948637  0.0613619  4.8053 1.545e-06 ***
## p 0.0294261  0.0097315  3.0238  0.002496 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## -2 log L: 276.2133

round(dnbinom(0:47, size, prob, log = FALSE), 4)
##  [1] 0.4111 0.1340 0.0834 0.0601 0.0463 0.0370 0.0303 0.0253 0.0214 0.0182
## [11] 0.0156 0.0135 0.0117 0.0103 0.0090 0.0079 0.0070 0.0061 0.0054 0.0048
## [21] 0.0043 0.0038 0.0034 0.0030 0.0027 0.0024 0.0022 0.0019 0.0017 0.0016
## [31] 0.0014 0.0013 0.0011 0.0010 0.0009 0.0008 0.0008 0.0007 0.0006 0.0006
## [41] 0.0005 0.0005 0.0004 0.0004 0.0003 0.0003 0.0003 0.0002
```

6.4.2.4 ML-Schätzer zur Normalverteilung

Funktion mle2() in library(bbmle)

```r
x    <- c(23, 25, 30, 18, 17, 24, 23, 20, 19)        # Beobachtungen

mean(x); sd(x)                                       # analytische ösung
## [1] 22.11111
## [1] 4.075673

library(bbmle)                                       # m=20 und s=4 initial
logL <- function(m=20, s=4) -sum(stats::dnorm(x, mean=m, sd=s, log=TRUE))
mle2(logL)
##
## Call:
## mle2(minuslogl = logL)
##
## Coefficients:
##         m         s
## 22.111221  3.842649
##
## Log-likelihood: -24.89
```

Funktion fitdistr() in library(MASS)

```
library(MASS)
x    <- c(23, 25, 30, 18, 17, 24, 23, 20, 19)        # Beobachtungen
fitdistr(x, "normal")                                # ML-Schätzung aus fitdistr()
##       mean          sd
##   22.1111111    3.8425814
##  ( 1.2808605) ( 0.9057051)
```

```
set.seed(123);
x <- rnorm(20, mean=80, sd=15)                       # Zufallszahlen
fitdistr(x, "normal")                                # ML-Schätzung
##      mean         sd
##   82.124357   14.220553
##  ( 3.179812) ( 2.248467)
```

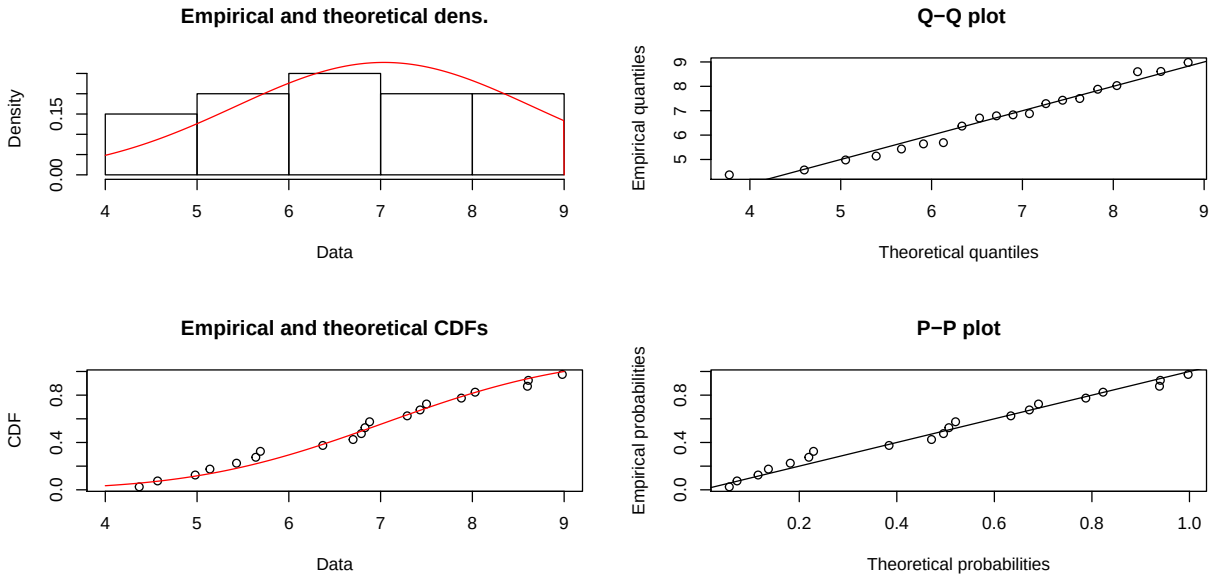
6.4.2.5 ML-Schätzer zur gestutzten Normalverteilung

Beispiel Feinstaub:

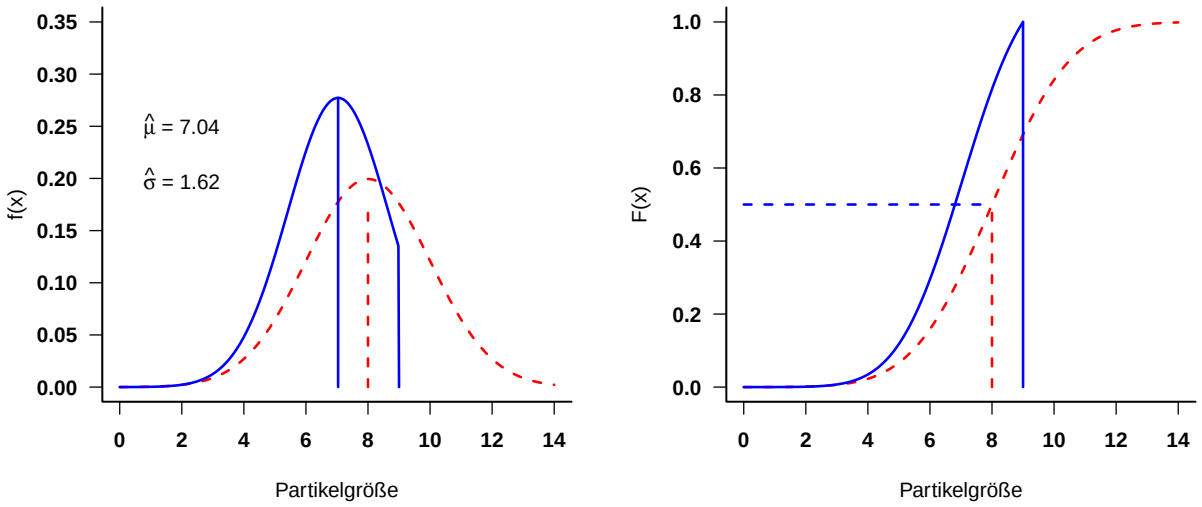
Funktion fitdist() in library(bbmle)

Funktion dtnorm() in library(extraDistr)

```
library(fitdistrplus); library(extraDistr)

filter <- c(4.98, 8.60, 6.37, 4.37, 8.03, 7.43, 6.83, 5.64, 5.43, 6.88,
            4.57, 7.50, 5.69, 7.88, 8.98, 6.79, 8.61, 6.70, 5.14, 7.29)

fit  <- fitdist(filter, dtnorm, fix.arg=list(a=-Inf, b=9),
                start=list(mean=mean(filter), sd=sd(filter)),
                optim.method="L-BFGS-B",
                lower=c(-0.1, -0.1), upper=c(Inf, Inf))
summary(fit)
## Fitting of the distribution ' tnorm ' by maximum likelihood
## Parameters :
##      estimate Std. Error
## mean 7.036644  0.5592341
## sd   1.622802  0.4099704
## Fixed parameters:
##   value
## a  -Inf
## b     9
## Loglikelihood:  -33.04198   AIC:  70.08397   BIC:  72.07543
## Correlation matrix:
##           mean        sd
## mean 1.0000000 0.6253291
## sd   0.6253291 1.0000000

plot(fit)
```



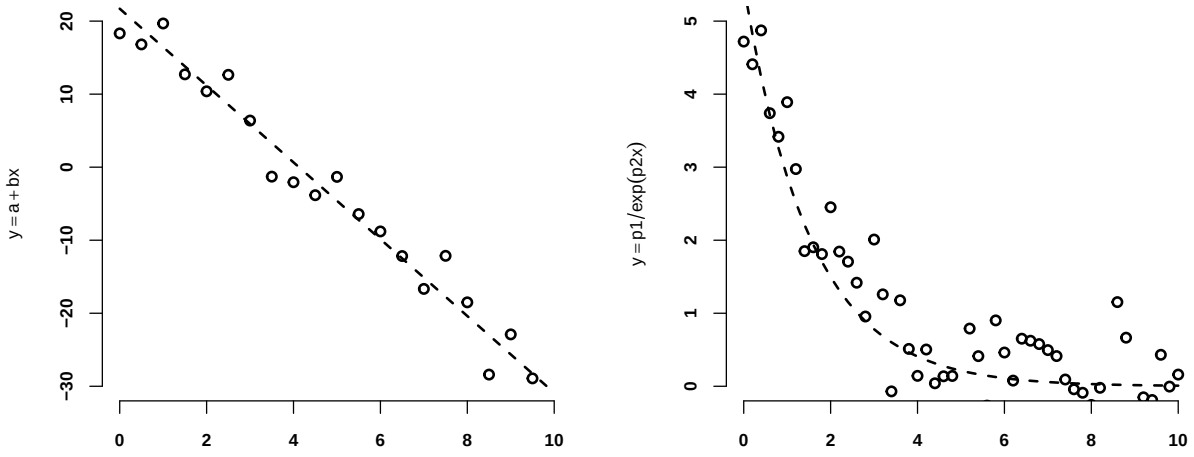

```
x     <- seq(0, 14, 0.02)
f1    <- dnorm(x, mean=8, sd=2)
f11   <- pnorm(x, mean=8, sd=2)
fitm  <- fit$estimate[1]
fits  <- fit$estimate[2]
f2    <- dtnorm(x, mean=fitm, sd=fits, a=0, b=9)
f22   <- ptnorm(x, mean=fitm, sd=fits, a=0, b=9)
mx    <- which(x == 9); trunc <- which(x == 9)

par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="l", ps=12)
plot(x, f1, las=1, type="l", lwd=2, lty=2,
     xlim=c(0,14), ylim=c(0,0.35), ylab="f(x)",
     xlab="Partikelgröße", col="red" )
lines(x[1:trunc], f2[1:trunc], lwd=2, lty=1, col="blue")
lines(c(8, 8), c(0, f1[mx]), col="red", lty=2, lwd=2)
lines(c(fitm, fitm), c(0, dtnorm(fitm, mean=fitm,
                sd=fits, a=0, b=9)), col="blue", lty=1, lwd=2)
text(2, 0.25, expression(paste(hat(mu), " = 7.04")))
text(2, 0.20, expression(paste(hat(sigma), " = 1.62")))

plot(x, f11, las=1, type="l", lwd=2, lty=2,
     xlim=c(0,14), ylim=c(0,1), ylab="F(x)",
     xlab="Partikelgröße", col="red" )
lines(x[1:trunc], f22[1:trunc], lwd=2, lty=1, col="blue")
lines(c(x[trunc],x[trunc]), c(0,1), lwd=2, col="blue")
lines(c(8, 8), c(0, 0.5), col="red", lty=2, lwd=2)
lines(c(0, 8), c(0.5, 0.5), col="blue", lty=2, lwd=2)
```



6.4.3 Schätzung nach dem keinsten Fehler (OLS)

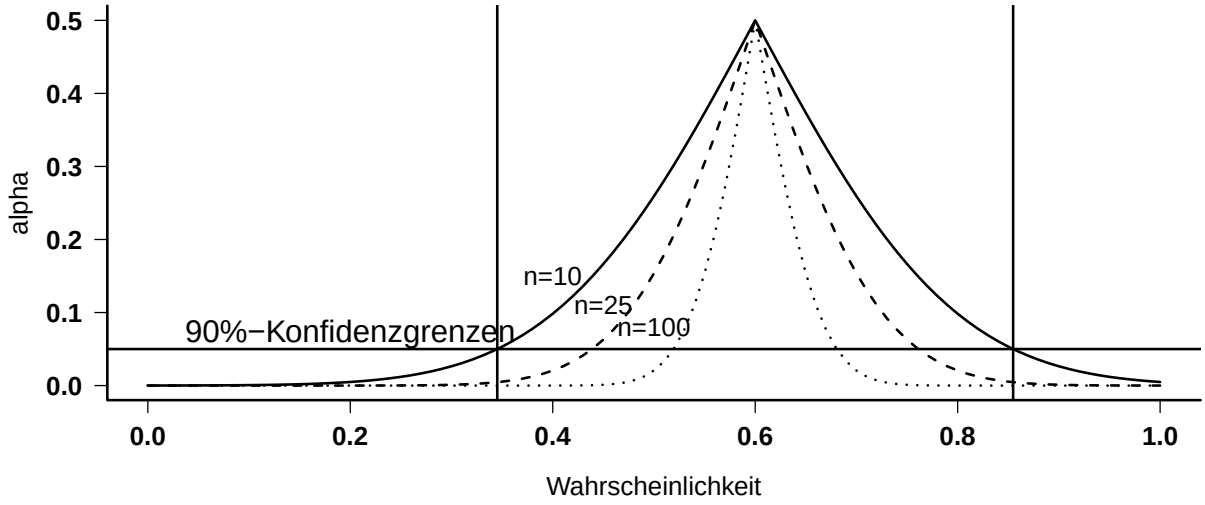
```
x1 <- seq(0, 10, by=0.5);  n1 <- length(x1)                    # lineares Modell
set.seed(123); e1 <- rnorm(n1, mean=0, sd=3)                   # Rauschen
y1 <- 20 - 5*x1 + e1                                           # Parameter a=20 und b=5
lm(y1 ~ x1)                                                    # Funktion lm()
##
## Call:
## lm(formula = y1 ~ x1)
##
## Coefficients:
## (Intercept)           x1
##      21.139       -5.177

x2 <- seq(0,10, by=0.2); n2 <- length(x2)                      # nichtlineares Modell
set.seed(123); e2 <- rnorm(n2, mean=0, sd=0.5)                 # Rauschen
y2 <- 5/exp(0.5*x2) + e2                                       # Parameter p1=5 und p2=0.5
nls(y2 ~ p1/exp(p2*x2), start=list(p1=1, p2=1))                # Funktion nls()
## Nonlinear regression model
##   model: y2 ~ p1/exp(p2 * x2)
##    data: parent.frame()
##     p1     p2
## 5.1611 0.5108
##  residual sum-of-squares: 10.45
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 2.434e-06

par(mfrow=c(1,2), lwd=2, font.axis=2, bty="n", ps=10)

plot(x1, y1, type="p", ylim=c(-30, 20), xlim=c(0,10),
```

```
                        ylab=expression(y==a+bx), xlab=" ")
lines(x1, 21.7 - 5.26 * x1, lty=2, cex=1.0, col="black")

plot(x2, y2, type="p", ylim=c(0,5), xlim=c(0,10),
                        ylab=expression(y==p1/exp(p2*x)), xlab=" ")
lines(x2, 5.49 / exp(0.65 * x2) , lty=2, col="black")
```



6.5 Intervallschätzung - Konfidenzintervalle

```
pi.hat  <- 0.6
n       <- 10
pi      <- seq(0, 1, 0.01)

confidence <- function(pi.hat, n, pi) {
  sd.hat <- sqrt(pi.hat*(1-pi.hat)/n)
  z      <- abs(pi.hat - pi)/sd.hat
  alpha  <- pnorm(z); 1-alpha        }

par(lwd=2, font.axis=2, bty="l", ps=15, mfrow=c(1,1))
plot(pi, confidence(pi.hat, 10, pi), las=1, type="l", lty=1,
     ylab="alpha", xlab="Wahrscheinlichkeit")
lines(pi, confidence(pi.hat, 25, pi), lty=2)
lines(pi, confidence(pi.hat, 100, pi), lty=3)
abline(h=0.05)
text(0.2, 0.07,"90%-Konfidenzgrenzen", cex=1.2)
abline(v=pi.hat-qnorm(0.95)* sqrt(pi.hat*(1-pi.hat)/n))
abline(v=pi.hat+qnorm(0.95)* sqrt(pi.hat*(1-pi.hat)/n))
text(0.40, 0.15, "n=10")
text(0.45, 0.11, "n=25")
text(0.50, 0.08, "n=100")
```

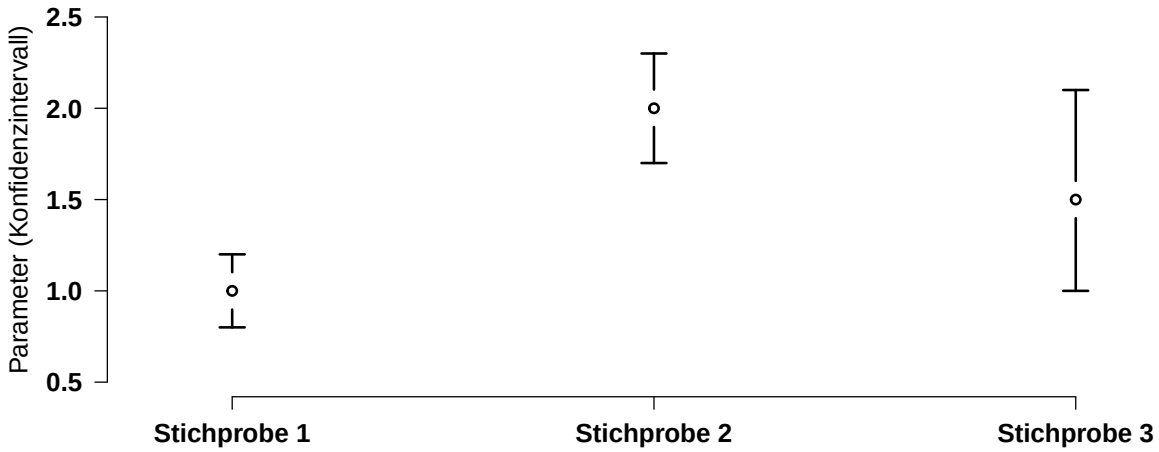


Vergleichende graphische Darstellung (Plot):

Funktion plotCI() in library(gplots)

```
library(gplots)
par <- c(1.0, 2.0, 1.5)
upp <- c(1.2, 2.3, 2.1); low <- c(0.8, 1.7, 1.0)
```

```r
par(mfrow=c(1,1), lwd=2, bty="n", font=1, font.axis=2, ps=15)

plotCI(par, ui=upp, li=low, xlim=c(0.8, 3.2), ylim=c(0.5, 2.5),
       las=1, xaxt="n",
         ylab="Parameter (Konfidenzintervall)", xlab=" ")
axis(side=1, at=1:3, labels=c("Stichprobe 1", "Stichprobe 2","Stichprobe 3"), cex=0.7)
```



6.6 Konfidenzintervall für Anteilswerte

Beispiel nach Fisher / Clopper-Pearson:

```r
n <- 20; x <- 7                                                    # Fisher-Verteilung
piu <- x/(x+(n-x+1)*qf(0.975, 2*(n-x+1), 2*x)); piu
## [1] 0.1539092
pio <- (x+1) * qf(0.975, 2*(x+1), 2*(n-x)) /
       (n-x+(x+1) * qf(0.975, 2*(x+1), 2*(n-x))); pio
## [1] 0.5921885

CI.Clopper <- function(x, n, conflev) {                # CI exakt nach Clopper-Pearson
           phat   <- x / n
           level  <- (1-conflev)/2
           lowlim <- qbeta(level, x, n-x+1)
           uplim  <- qbeta(1-level, x+1, n-x)
cat(" \nCI Clopper / Pearson: Mit Überdeckungswahrscheinlichkeit ",conflev,
    " \nund der Schätzung", round(phat,digits=4), "ergibt sich für die untere und",
    " \nobere Vertrauensgrenze:", round(lowlim,digits=4),"-",
round(uplim,digits=4)," \n")          }

CI.Clopper(7, 20, 0.95)
##
## CI Clopper / Pearson: Mit Überdeckungswahrscheinlichkeit  0.95
```

```
## und der Schätzung 0.35 ergibt sich für die untere und
## obere Vertrauensgrenze: 0.1539 - 0.5922
```

Beispiel Prävalenzschätzung:

```
x <- 100;   k <- 20
conflev <- 0.95; level  <- (1-conflev)/2
phat    <- (k-1) / (x+k-1); round(phat, 3)
## [1] 0.16
lowlim  <- qbeta(level, k, x+1); round(lowlim, 3)
## [1] 0.105
uplim   <- qbeta(1-level, k, x); round(uplim, 3)
## [1] 0.238
```

6.6.1 Approximationen durch die Normalverteilung

```
CI.Binom <- function(x, n, conflev) {
        phat   <- x / n
        zalpha <- abs(qnorm((1-conflev)/2))                    # KI Wald-Statistik
        bound  <- 1/(2*n) + zalpha*sqrt(phat*(1-phat)/n)
        low1   <- phat - bound; upp1   <- phat + bound
                                                               # KI Wilson-Intervall
        midpnt <- (x + (zalpha**2/2))/(n + zalpha**2)
        bound  <- zalpha*(sqrt(n)/(n+zalpha**2))*
                    sqrt(phat*(1-phat)+zalpha**2/(4*n))
        upp2   <- midpnt + bound;       low2   <- midpnt - bound
cat("Mit Überdeckungswahrscheinlichkeit ", conflev,"und der Schätzung",
round(phat,digits=4),"\nist die angenäherte untere und obere Vertrauensgrenze nach",
"\n Wald-Statistik  :",round(low1,digits=4),"-", round(upp1,digits=4),
"\n Wilson-Intervall:",round(low2,digits=4),"-", round(upp2,digits=4)," \n") }

CI.Binom(70, 200, 0.95)
## Mit Überdeckungswahrscheinlichkeit  0.95 und der Schätzung 0.35
## ist die angenäherte untere und obere Vertrauensgrenze nach
##  Wald-Statistik  : 0.2814 - 0.4186
##  Wilson-Intervall: 0.2873 - 0.4184
CI.Binom(7, 20, 0.95)
## Mit Überdeckungswahrscheinlichkeit  0.95 und der Schätzung 0.35
## ist die angenäherte untere und obere Vertrauensgrenze nach
##  Wald-Statistik  : 0.116 - 0.584
##  Wilson-Intervall: 0.1812 - 0.5671
```

6.6.4 Konfidenzintervall für die Differenz zweier Anteile

```
ci.wald <- function(x1, n1, x2, n2, conflev) {                # Standardverfahren (Wald)
           zalpha <- abs(qnorm((1-conflev)/2))
           p1 <- x1/n1; p2 <- x2/n2;           delta  <- p1 - p2
           adj    <- ifelse(delta<0, 0.5*(1/n1 + 1/n2), -0.5*(1/n1 + 1/n2))
           bound  <- zalpha*sqrt(p1*(1-p1)/n1 + p2*(1-p2)/n2)
           lowlim <- delta + adj - bound;      upplim <- delta + adj + bound
cat(" \nCI nach Wald: Für das", 100*conflev, "%-Konfidenzintervall zu der Schätzung",
round(p1 - p2,digits=4), "\nergibt sich die untere und obere Vertrauensgrenze:",
round(lowlim,digits=4),"-", round(upplim,digits=4)," \n")
}

ci.wald(140, 200, 150, 250, 0.95)
##
## CI nach Wald: Für das 95 %-Konfidenzintervall zu der Schätzung 0.1
## ergibt sich die untere und obere Vertrauensgrenze: 0.0076 - 0.1834
```

Wilson-Scores:

```
ci.wilson <- function(x1, n1, x2, n2, conflev) {          # Wilson-Score
           z  <- abs(qnorm((1-conflev)/2))
           p1 <- x1/n1; p2 <- x2/n2
           phi1 <- (2*x1 + z^2)/(2*(n1 + z^2)); phi2 <- (2*x2 + z^2)/(2*(n2 + z^2))
           psi1 <- x1^2/(n1^2 + n1*z^2);        psi2 <- x2^2/(n2^2 + n2*z^2)
           l1   <- phi1 - sqrt(phi1^2 - psi1);  l2   <- phi2 - sqrt(phi2^2 - psi2)
           u1   <- phi1 + sqrt(phi1^2 - psi1);  u2   <- phi2 + sqrt(phi2^2 - psi2)
           delta <- sqrt((p1 - l1)^2 + (u2 - p2)^2)
           epsil <- sqrt((u1 - p1)^2 + (p2 - l2)^2)
           lowlim <- (p1 - p2) - delta;         upplim <- (p1 - p2) + epsil
cat(" \nCI nach Wilson: Für das", 100*conflev, "%-Konfidenzintervall zu der Schätzung",
round(p1 - p2,digits=4), "\nergibt sich die untere und obere Vertrauensgrenze:",
round(lowlim,digits=4),"-", round(upplim,digits=4)," \n")
}

ci.wilson(140, 200, 150, 250, 0.95)
##
## CI nach Wilson: Für das 95 %-Konfidenzintervall zu der Schätzung 0.1
## ergibt sich die untere und obere Vertrauensgrenze: 0.011 - 0.1856
```

6.6.4 Konfidenzintervall für das Verhältnis zweier Anteile

```
ci.ratio <- function(x1, n1, x2, n2, conflev) {
   z   <- abs(qnorm((1-conflev)/2))
   p1  <- x1/n1; p2 <- x2/n2; rr  <- p1/p2
                                              # Adjusted Wald-based method
   low1 <- (n2/n1)*((x1+0.5)/(x2-0.5))*exp(-z*sqrt(1/(x2+0.5)+1/(x1+0.5)))
   upp1 <- (n2/n1)*((x1+0.5)/(x2-0.5))*exp(z*sqrt(1/(x2+0.5)+1/(x1+0.5)))
                                              # Score Method (Wilson)
   x. <- x1 + x2; p.hat <- x2/x.
   p.l <- (x./(x.+z^2))*(p.hat+(z^2/(2*x.)) +
                  z*sqrt((1/x.)*(p.hat*(1-p.hat)+z^2/(4*x.))))
   p.u <- (x./(x.+z^2))*(p.hat+(z^2/(2*x.)) -
                  z*sqrt((1/x.)*(p.hat*(1-p.hat)+z^2/(4*x.))))
   low2 <- n2*(1-p.l)/(n1*p.l);    upp2 <- n2*(1-p.u)/(n1*p.u)
cat("\nDas",100*conflev,"%-KI für das Verhältnis",round(rr,digits=2),"zweier Anteile",
 "\nergibt sich aus der unteren und oberen Vertrauensgrenze nach",
 "\nWald-Statistik (adj.):",round(low1,digits=4),"-", round(upp1,digits=4),
 "\nWilson               :",round(low2,digits=4),"-", round(upp2,digits=4),"\n")
}
```

Beispiel Likelihood-Quotient:

```
ci.ratio(30, 40, 5, 50, 0.95)
##
## Das 95 %-KI für das Verhältnis 7.5 zweier Anteile
## ergibt sich aus der unteren und oberen Vertrauensgrenze nach
## Wald-Statistik (adj.): 3.4172 - 21.0049
## Wilson               : 3.0052 - 18.7173
```

Beispiel relatives Risiko:

```
ci.ratio(11, 219, 1, 183, 0.95)
##
## Das 95 %-KI für das Verhältnis 9.19 zweier Anteile
## ergibt sich aus der unteren und oberen Vertrauensgrenze nach
## Wald-Statistik (adj.): 3.5059 - 105.3599
## Wilson               : 1.5257 - 55.3777
```

6.6.6 Mindestumfang einer Stichprobe zur Schätzung eines Anteils

Tabelle:

```
alpha <- 0.05                                                    # Konfidenz
zalph <- qnorm(1-alpha/2)                                       # Quantilgrenzen (zweiseitig)
preci <- seq(0.01, 0.10, by=0.01)
width <- 2*preci; lw  <- length(width)
prob  <- c(0.01,seq(0.02, 0.10, by=0.02),seq(0.15, 0.50, by=0.05))
lp    <- length(prob)

ntab  <- matrix(rep(NA, lp*lw), ncol = lp, byrow = TRUE)
for (i in 1:lw) {
  for (j in 1:lp) ntab[i,j] <- ceiling(((2*zalph)/width[i])^2 * prob[j] * (1-prob[j])) }
ntab <- cbind(seq(0.01, 0.1, 0.01), preci, ntab)
colnames(ntab) <- c("+/-w","p","0.01","0.02","0.04","0.06","0.08","0.10",
                    "0.15","0.20","0.25","0.30","0.35","0.40","0.45","0.50")
as.data.frame(ntab[, 1:14])
```

+/-w	p	0.01	0.02	0.04	0.06	0.08	0.10	0.15	0.20	0.25	0.30	0.35	0.40
0.01	0.01	381	753	1476	2167	2828	3458	4898	6147	7203	8068	8740	9220
0.02	0.02	96	189	369	542	707	865	1225	1537	1801	2017	2185	2305
0.03	0.03	43	84	164	241	315	385	545	683	801	897	972	1025
0.04	0.04	24	48	93	136	177	217	307	385	451	505	547	577
0.05	0.05	16	31	60	87	114	139	196	246	289	323	350	369
0.06	0.06	11	21	41	61	79	97	137	171	201	225	243	257
0.07	0.07	8	16	31	45	58	71	100	126	147	165	179	189
0.08	0.08	6	12	24	34	45	55	77	97	113	127	137	145
0.09	0.09	5	10	19	27	35	43	61	76	89	100	108	114
0.10	0.10	4	8	15	22	29	35	49	62	73	81	88	93

Beispiel Fersehprogramm:

```
nprobN <- function(N, w, p, alpha) {
    zalpha <- qnorm(1-alpha/2); i <- w/2
    return(ceiling((N*(i/zalpha)^2 + N*p - N*p^2)/
                   (N*(i/zalpha)^2 + p - p^2)))     }

nprobN(1000, 0.20, 0.50, 0.05)
## [1] 89

nprobN(1000, 0.20, 0.30, 0.05)
## [1] 76
```

Tabelle (Variationskoeffizient fest vorgegeben):

```
p  <- c(seq(0.5, 0.05, by=-0.05), 0.04, 0.03,0.02,0.01); lp <- length(p)
cv <- c(0.10, 0.15, 0.20, 0.25); lcv <- length(cv)

n  <- matrix(rep(NA, lp*lcv), ncol = lcv, byrow = TRUE)
for (i in 1:lp) {
```

```
    for (j in 1:lcv) n[i,j] <- ceiling((1-p[i]) / (p[i]*cv[j]^2)) }
tab <- as.data.frame(cbind(p, n))
colnames(tab) <- c("p","10%","15%","20%","25%")
as.data.frame(tab)
```

p	10%	15%	20%	25%
0.50	100	45	25	16
0.45	123	55	31	20
0.40	150	67	38	24
0.35	186	83	47	30
0.30	234	104	59	38
0.25	300	134	75	48
0.20	400	178	100	65
0.15	567	252	142	91
0.10	900	401	225	145
0.05	1900	845	475	304
0.04	2400	1067	600	384
0.03	3234	1438	809	518
0.02	4900	2178	1225	784
0.01	9900	4400	2475	1584

6.6.7 Simultane Konfidenzintervalle für multinomiale Anteile

```
CImultinom <- function(n, alpha=0.05) {                    # nach L.A. Goodman
    k <- length(n); N <- sum(n)
    cil <- rep(NA, k); ciu <- rep(NA, k)
    q   <- qchisq(1-alpha/k, df=1)
    for (i in 1:k) {
        T <- sqrt(q * (q + 4*n[i]*(N-n[i])/N))
        cil[i] <- (q + 2*n[i] - T) / (2*(N+q))
        ciu[i] <- (q + 2*n[i] + T) / (2*(N+q))
    }
    round(cbind(cil, p=n/N, ciu), 4)
}

xmpl <- c(56,72,73,59,62)
CImultinom(xmpl)
##         cil      p    ciu
## [1,] 0.1262 0.1739 0.2348
## [2,] 0.1697 0.2236 0.2886
## [3,] 0.1725 0.2267 0.2920
## [4,] 0.1343 0.1832 0.2450
## [5,] 0.1424 0.1925 0.2551
```

Funktion multinomialCI() in library(MultinomialCI)

```
library(MultinomialCI)
multinomialCI(xmpl, alpha=0.05, verbose=F)
##           [,1]      [,2]
```

```
## [1,] 0.1211180 0.2320627
## [2,] 0.1708075 0.2817521
## [3,] 0.1739130 0.2848577
## [4,] 0.1304348 0.2413795
## [5,] 0.1397516 0.2506962
```

Fallzahlabschätzung:

```
nCImultinom <- function(k, d, alpha=0.05) {
    f <- rep(NA, k)
    for (m in 1:k) { z <- qnorm(p=1-alpha/(2*m))
                     f[m] <- (z^2*(1/m)*(1-1/m))/d^2 }
    ceiling(max(f))                          }

nCImultinom(k=5, d=0.06, alpha=0.05)
## [1] 354
```

6.7 Konfidenzintervalle für den Erwartungswert einer Poisson-Verteilung

```
ci.poisson <- function(k, n, conf.level=0.95) {
  alpha  <- 1 - conf.level
  lambda <- k/n
  l_l <- 1/(2*n) * qchisq(alpha/2, df=2*k)
  l_u <- 1/(2*n) * qchisq(1-alpha/2, df=2*k+2)
  cat(" \n Das",conf.level,"Konfidenzintervall für Lambda =",
      round(lambda, 4)," ist [",
      round(l_l, 4),"-",round(l_u, 4),"] \n")
}
```

Beispiel Vogelnester:

```
n <- 40                                                    # Areale
k <- 44                                                    # Nester
ci.poisson(k=44, n=40, conf.level=0.95)
##
##  Das 0.95 Konfidenzintervall für Lambda = 1.1  ist [ 0.7993 - 1.4767 ]
```

6.7.3 Konfidenzintervall für das Verhältnis zweier Raten

```
ci.rate.pois <- function(k1, n1, k2, n2, conf.level=0.95) {
  alpha  <- 1 - conf.level
  lambda1 <- k1/n1; lambda2 <- k2/n2
  k <- k1; m <- k1+k2
  F  <- qf(0.975, 2*(m-k+1), 2*k); pl <- k/(k + (m-k+1)*F)
  F  <- qf(0.975, 2*(k+1), 2*(m-k)); pu <- (k+1)*F/(m-k+(k+1)*F)
  l_l <- n2*pl/(n1*(1-pl))
  l_u <- n2*pu/(n1*(1-pu))
  cat(" \n Das",conf.level,"Konfidenzintervall für das Verhältnis von",
      round(lambda1, 4),"zu",round(lambda2, 4),"\n ist [",
      round(l_l, 4),"-",round(l_u, 4),"] \n")
}
ci.rate.pois(40, 20, 22, 30, conf.level=0.95)
##
##  Das 0.95 Konfidenzintervall für das Verhältnis von 2 zu 0.7333
##  ist [ 1.5824 - 4.8181 ]
```

Funktion poisson.exact() in library(exactci)

```
library(exactci)
poisson.exact(c(40, 22), c(20, 30))
##
##  Exact two-sided Poisson test (central method)
##
## data:  c(40, 22) time base: c(20, 30)
```

```
## count1 = 40, expected count1 = 24.8, p-value = 0.0001669
## alternative hypothesis: true rate ratio is not equal to 1
## 95 percent confidence interval:
##  1.582402 4.818070
## sample estimates:
## rate ratio
##   2.727273
```

Beispiel Ausfallraten zweier Stromleitungen:

```
k1 <- 69; n1 <- 1079.6;  k2 <- 12; n2 <- 467.9
l1 <- k1/n1; l2 <- k2/n2;
r.hat = l2/l1; r.hat
## [1] 0.4012749
s.hat <- sqrt(1/k1+1/k2); s.hat
## [1] 0.3127716
l_l <- r.hat/exp(1.96*s.hat); l_u <- r.hat*exp(1.96*s.hat)
round(l_l, 2); round(l_u, 2)
## [1] 0.22
## [1] 0.74

ci.rate.pois(12, 467.9, 69, 1079.6, conf.level=0.95)
##
##  Das 0.95 Konfidenzintervall für das Verhältnis von 0.0256 zu 0.0639
##  ist [ 0.1978 - 0.7467 ]
```

6.7.4 Konfidenzintervalle für standardisierte Raten

```
age <- c("0-19","20-44","45-64","65 und älter")              # Raten in der Medizin
d.i <- c(   34,   135,   299,  1167); sum(d.i)
## [1] 1635
n.i <- c(34000, 75500, 27000, 15000); sum(n.i)
## [1] 151500

sum(d.i)/sum(n.i)                                            # rohe Rate
## [1] 0.01079208

s.i  <- c( 0.18,  0.40,  0.25,  0.17)                        # Referenz-Population
N    <- 1000000
N.i  <- N*s.i

rate <- sum(d.i * (s.i/n.i)); rate                           # altersstand. Rate
## [1] 0.01688975
var <- sum(d.i *(s.i/n.i)^2); var                            # Varianzschätzung
## [1] 1.802713e-07

r.i   <- c(0.0005, 0.002, 0.008, 0.065)
e.i   <- r.i * n.i; sum(e.i)
## [1] 1359
tab <- cbind(age, N.i, n.i, d.i, r.i, e.i)
colnames(tab) <- c("Alter","ref.Pop.","Population","Todesfälle","ref.Rate","geschätzt")
as.data.frame(tab)
```

Alter	ref.Pop.	Population	Todesfälle	ref.Rate	geschätzt
0-19	180000	34000	34	5e-04	17
20-44	4e+05	75500	135	0.002	151
45-64	250000	27000	299	0.008	216
65 und älter	170000	15000	1167	0.065	975

```
alpha  <- 0.05
dfl    <- (2*rate^2)/var
lowlim <- (var/(2*rate)) * qchisq(alpha/2, dfl)
maxwt  <- max(s.i/n.i)
dfu    <- 2*(rate+maxwt)^2/(var+maxwt^2)
upplim <- (var + maxwt^2)/(2*(rate + maxwt)) * qchisq(1-alpha/2, dfu)
lowlim; upplim
## [1] 0.01606774
## [1] 0.01774361
```

Standardisiertes Mortalitätsverhältnis (SMR):

```
alpha <- 0.05; z <- qnorm(1-alpha/2)
r.i   <- c(0.0005, 0.002, 0.008, 0.065)
e.i   <- r.i * n.i; sum(e.i)
## [1] 1359
O     <- sum(d.i); E <- sum(e.i)
SMR   <- (O/E)*100; SMR
## [1] 120.3091
lowlim <- ((O/E)*100)   * (1 - 1/(9*O)     - z/(3*sqrt(O)))^3
upplim <- ((O+1)/E)*100 * (1 - 1/(9*(O+1)) + z/(3*sqrt(O+1)))^3
lowlim; upplim
## [1] 114.5474
## [1] 126.2854
```

Fallzahl: Erkennbare Risiken mit fester Power:

```
rpwr.risk <- function(E, R=NULL, alpha=0.05, power=NULL) {
  beta <- 1 - power
  if ((is.null(power) & is.null(R)))  stop
  if(is.null(power)) power <- pnorm(2*sqrt(E)*(sqrt(R)-1) - qnorm(1-alpha))
  if(is.null(R))  R <- (1 + (qnorm(1-alpha)+qnorm(1-beta))/(2*sqrt(E)))^2
  cat(" Erwartete Fälle :", E,"\n",
      "Signif.-Niveau  :", alpha,"\n",
      "Power           :", round(power, 4), "\n",
      "Rel. Risk       :", round(R,4), "\n")
}

rpwr.risk(E=c(1,5,10,15,20,25,30), power=0.80)
##  Erwartete Fälle : 1 5 10 15 20 25 30
##  Signif.-Niveau  : 0.05
##  Power           : 0.8
##  Rel. Risk       : 5.0321 2.4211 1.9409 1.745 1.6333 1.5591 1.5055
```

6.8 Konfidenzintervalle für den Erwartungswert bei Normalverteilung

8.8.2 Konfidenzintervall für den Erwartungswert

```
x <- c(95,  84, 105,  96,  86,  86,  95,  94,  75,  93)
n <- length(x)
m <- mean(x); m
## [1] 90.9
s <- sd(x); s
## [1] 8.305955
m - qt(0.975, n-1)*s/sqrt(n)                                    # untere Vertrauensgrenze
## [1] 84.95828

m + qt(0.975, n-1)*s/sqrt(n)                                    # obere Vertrauensgrenze
## [1] 96.84172

t.test(x, mu = 90, conf.level = 0.95)                           # Funktion t.tes()
##
##  One Sample t-test
##
## data:  x
## t = 0.34265, df = 9, p-value = 0.7397
## alternative hypothesis: true mean is not equal to 90
## 95 percent confidence interval:
##  84.95828 96.84172
## sample estimates:
## mean of x
##      90.9
```

6.8.4 Konfidenzintervall für den Erwartungswert aus Paardifferenzen

```
x <- c(4.0, 3.5, 4.1, 5.5, 4.6, 6.0, 5.1, 4.3)
y <- c(3.0, 3.0, 3.8, 2.1, 4.9, 5.3, 3.1, 2.7)
d <- x - y; d
## [1]  1.0  0.5  0.3  3.4 -0.3  0.7  2.0  1.6

t.test(x, y, mu=0, paired=TRUE, con.level = 0.95)               # Funktion t.test()
##
##  Paired t-test
##
## data:  x and y
## t = 2.798, df = 7, p-value = 0.0266
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1781177 2.1218823
## sample estimates:
## mean of the differences
##                    1.15
```

6.8.7 Konfidenzintervall für den Erwartungswert einer Lognormalverteilung

Beispiel Kohlenmonoxid:

```
x.CO <- c(12.5, 20, 4, 20, 25, 170, 15, 20, 15)
n <- length(x.CO);  y.CO <- log(x.CO)

x.mean <- mean(x.CO); x.stdv <- sd(x.CO); x.med <- median(x.CO)
print(paste("CO-Wert (X)",x.mean, x.med, x.stdv))
## [1] "CO-Wert (X) 33.5 20 51.5351821574349"

y.mean <- mean(y.CO); y.stdv <- sd(y.CO); y.med <- median(y.CO)
print(paste("Y=log(X)   ",y.mean, y.med, y.stdv))
## [1] "Y=log(X)    2.96333272113478 2.99573227355399 0.974483478872704"
                                                          # naive Schätzung
lower_naive <- round(exp(y.mean - qt(0.975, (n-1)) * y.stdv/sqrt(n)), 2)
upper_naive <- round(exp(y.mean + qt(0.975, (n-1)) * y.stdv/sqrt(n)), 2)
print(paste(lower_naive,"<=", round(exp(y.mean), 2),"<=", upper_naive))
## [1] "9.15 <= 19.36 <= 40.95"
                                                        # Schätzung nach Cox
m         <- y.mean + y.stdv^2/2
v         <- y.stdv^2/n + y.stdv^4/(2*(n-1))
lower_cox <- round(exp(m - qt(0.975, (n-1)) * sqrt(v)), 2)
upper_cox <- round(exp(m + qt(0.975, (n-1)) * sqrt(v)), 2)
print(paste(lower_cox,"<=", round(exp(m), 2),"<=",  upper_cox))
## [1] "12.31 <= 31.13 <= 78.72"
```

6.9 Konfidenzintervalle für die mittlere absolute Abweichung

```
x     <- c(10, 15, 20, 16, 13, 12, 15, 21, 11, 24, 17, 14, 12, 10, 30)
n     <- length(x)
medi  <- median(x)
c     <- n / (n-1)
tau.h <- sum(abs(x-medi))/n; tau.h*c
## [1] 4.357143
d     <- (mean(x) - medi)/tau.h;  g   <- var(x) / tau.h^2

varln.tau <- (d^2 + g -1)/n
upper <- exp(log(tau.h*c) + qnorm(0.975)*sqrt(varln.tau)); upper
## [1] 7.203192
lower <- exp(log(tau.h*c) - qnorm(0.975)*sqrt(varln.tau)); lower
## [1] 2.635595
```

6.10 Konfidenzintervalle für den Median

Tabelle:

```
alpha <- 0.05;  z <- qnorm(1-alpha/2)
n  <- 5 : 104
nl <- length(n)

hu <- round(n/2 - z*sqrt(n)/2)
ho <- round(1 + n/2 + z*sqrt(n)/2)
tab <- cbind(n, hu, ho)

tabout <- cbind(tab[1:20,], tab[21:40,], tab[41:60,], tab[61:80,],
                tab[81:100,])
colnames(tabout) <- rep(c("n","U","O"), 5)
as.data.frame(tabout)
```

n	U	O	n	U	O	n	U	O	n	U	O	n	U	O
5	0	6	25	8	18	45	16	30	65	25	41	85	33	53
6	1	6	26	8	19	46	16	31	66	25	42	86	34	53
7	1	7	27	8	20	47	17	31	67	25	43	87	34	54
8	1	8	28	9	20	48	17	32	68	26	43	88	35	54
9	2	8	29	9	21	49	18	32	69	26	44	89	35	55
10	2	9	30	10	21	50	18	33	70	27	44	90	36	55
11	2	10	31	10	22	51	19	33	71	27	45	91	36	56
12	3	10	32	10	23	52	19	34	72	28	45	92	37	56
13	3	11	33	11	23	53	19	35	73	28	46	93	37	57
14	3	12	34	11	24	54	20	35	74	29	46	94	37	58
15	4	12	35	12	24	55	20	36	75	29	47	95	38	58
16	4	13	36	12	25	56	21	36	76	29	48	96	38	59
17	4	14	37	13	25	57	21	37	77	30	48	97	39	59
18	5	14	38	13	26	58	22	37	78	30	49	98	39	60
19	5	15	39	13	27	59	22	38	79	31	49	99	40	60
20	6	15	40	14	27	60	22	39	80	31	50	100	40	61
21	6	16	41	14	28	61	23	39	81	32	50	101	41	61
22	6	17	42	15	28	62	23	40	82	32	51	102	41	62
23	7	17	43	15	29	63	24	40	83	33	51	103	42	62
24	7	18	44	15	30	64	24	41	84	33	52	104	42	63

```
ci_median <- function(x, alpha=0.05) {
  z <- qnorm(1-alpha/2);   n <- length(x)
  xmed <- median(x, na.rm = TRUE)
  hu <- round(n/2 - z*sqrt(n)/2); ho <- round(1 + n/2 + z*sqrt(n)/2)
  lu <- sort(x)[hu];              lo <- sort(x)[ho]
  cat("\n",(1-alpha)*100,"%-Konfidenzintervall","\n",
      "zum Median ",xmed,":"," (",lu,", ",lo,")","\n")
}

data <- c(12, 14, 10, 20, 9, 15, 22, 18, 26, 13, 27, 8, 10)
ci_median(data)
##
```

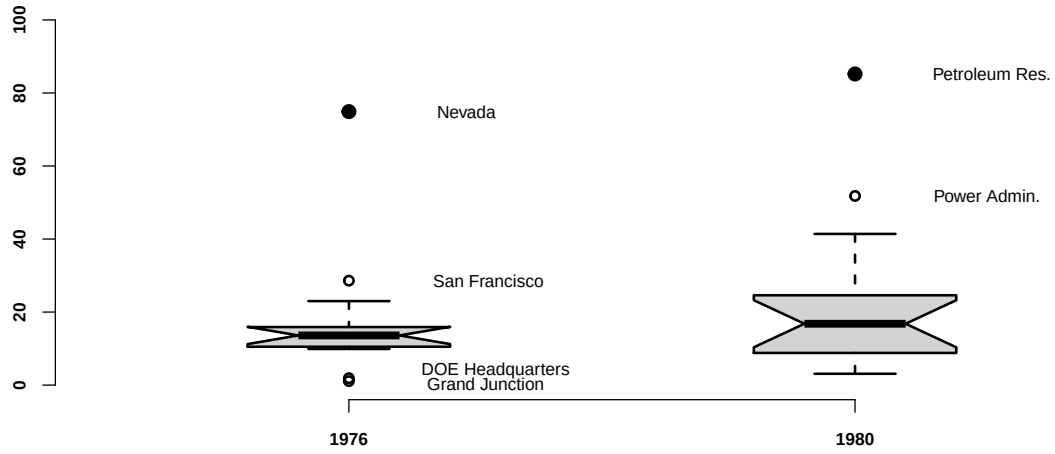
```
##  95 %-Konfidenzintervall
##  zum Median  14 :  ( 10 ,  22 )
```

Beispiel Energieversorger:

```
y1 <- c(15.9,13.6,1.9,1.1,13.8,23.0,28.6,15.0,10.5,
                    12.0,11.8,9.9,74.9,NA,NA)
y2 <- c(3.1, 5.2, 6.3,7.7,9.9,13.1,15.3,16.8,22.2,
                    22.3,23.3,25.9,41.4,51.8,85.2)
name  <- c("Schenectady N.R.","Savannah River","DOE Headquarters",
           "Grand Junction","Albuquerque","Oak Ridge","San Francisco",
           "Energy Tech Centers","Richland","Pittsburg N.R.",
           "Chicago","Idaho","Nevada","Power Admin.",
           "Petroleum Resources")
ausfall <- cbind(y1, y2);
colnames(ausfall) <- c("1976","1980"); rownames(ausfall) <- name
bwp <- boxplot(y1, y2, notch=TRUE, names=c("1976","1980"),
       pars = list(boxwex =0.5, staplewex=0.5, outwex=0.5), plot=FALSE)
t1  <- bwp$stats; colnames(t1) <- c("1976","1980")
rownames(t1) <- c("whisker low","1. quartil","median value",
                  "3. quartil","whisker high")
t2 <- bwp$conf; colnames(t1) <- c("1976","1980")
rownames(t2) <- c("95%-KI lower","95%-KI upper")
tab <- rbind(t1, t2)
as.data.frame(tab)
```

	1976	1980
whisker low	9.90000	3.10000
1. quartil	10.50000	8.80000
median value	13.60000	16.80000
3. quartil	15.90000	24.60000
whisker high	23.00000	41.40000
95%-KI lower	11.23365	10.35432
95%-KI upper	15.96635	23.24568

```
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="n", ps=10)
boxplot(y1, y2, notch=TRUE, names=c("1976","1980"),
       pars = list(boxwex =0.4, staplewex=0.4, outwex=0.4),
       ylim=c(0,100), col="lightgrey")
text(1.28, 74.9, "Nevada          ")
text(1.29, 28.6, "San Francisco   ")
text(1.29, 4,    "DOE Headquarters")
text(1.28, 0.5,  "Grand Junction  ")
text(2.28, 85.2, "Petroleum Res.  ")
text(2.28, 51.8, "Power Admin.    ")
points(1, 74.9, pch=20, cex=2)
points(2, 85.2, pch=20, cex=2)
```




```r
x <- c(95,  84, 105,  96,  86,  86,  95,  94,  75,  93)

wilcox.test(x, mu = 0, conf.int = TRUE, conf.level = 0.95)    # Funktion wilcox.test()
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  x
## V = 55, p-value = 0.005857
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##  84.99998 95.50002
## sample estimates:
## (pseudo)median
##       90.50004
```

6.10.1 Konfidenzintervall für die Differenz / den Quotient von Medianen

```r
mediconfi <- function(x, y, alpha=0.05) {
    zalpha <- qnorm(1-alpha/2)
    x  <- sort(x);                        y <- sort(y)
    nx <- length(x);                     ny <- length(y)
    xmed <- median(x);                 ymed <- median(y)
    cx <- round((nx +1)/2 - sqrt(nx)); cy <- round((ny +1)/2 - sqrt(ny))
    px <- 0;                              py <- 0
    for (i in 0:(cx-1)) px <- px + choose(nx, i) * 0.5^(nx)
    for (i in 0:(cy-1)) py <- py + choose(ny, i) * 0.5^(ny)
    zx <- qnorm(1-px); zy <- qnorm(1-py)
    varxmed  <- ((x[nx-cx+1]-x[cx])/(2*zx))^2
    varymed  <- ((y[ny-cy+1]-y[cy])/(2*zy))^2
    varxmeds <- ((log(x[nx-cx+1])-log(x[cx]))/(2*zx))^2
    varymeds <- ((log(y[ny-cy+1])-log(y[cy]))/(2*zy))^2
```

```
    mediquot <- round(xmed / ymed, 3)
    lmeddiff <- round(mediquot - zalpha * sqrt(varxmed + varymed), 3)
    umeddiff <- round(mediquot + zalpha * sqrt(varxmed + varymed), 3)
    lmedquot <- round(exp(log(xmed/ymed)-zalpha*(sqrt(varxmeds) + varymeds)), 3)
    umedquot <- round(exp(log(xmed/ymed)+zalpha*(sqrt(varxmeds) + varymeds)), 3)
    cat("\n",(1-alpha)*100,"%-Konfidenzintervall","\n",
        "zur Differenz ",xmed,"-",ymed,": = ",mediquot,
        " (",lmeddiff,", ",umeddiff,")","\n",
        "zum Quotienten",xmed,"/",ymed,": = ",mediquot,
        " (",lmedquot,", ",umedquot,")","\n")
}
```

Beispiel Brombeeren:

```
sonnig   <- c(4.1, 4.5, 4.8, 5.1, 5.1, 5.3, 5.5, 6.0)
schattig <- c(5.5, 5.5, 5.5, 5.9, 6.3, 6.5, 6.8, 7.2)
quantile(sonnig); quantile(schattig)
##    0%   25%   50%   75%  100%
## 4.100 4.725 5.100 5.350 6.000
##    0%   25%   50%   75%  100%
## 5.500 5.500 6.100 6.575 7.200

mediconfi(sonnig, schattig, alpha=0.05)
##
##  95 %-Konfidenzintervall
##  zur Differenz  5.1 - 6.1 : =  -1  ( -1.888 ,  -0.112 )
##  zum Quotienten 5.1 / 6.1 : =  0.836  ( 0.745 ,  0.938 )
                                                  # Differenz: Funktion wilcox.test()
wilcox.test(sonnig, schattig, alternative = "two.sided",
            correct = TRUE, conf.int = TRUE)
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  sonnig and schattig
## W = 5.5, p-value = 0.005907
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -1.8000025 -0.3999249
## sample estimates:
## difference in location
##              -1.025371
```

6.10.2 Verteilungsunabhängige Konfidenzintervalle für beliebige Quantile

```
ci.perc <- function(data, p, level) {
  n <- length(data);  plev <- 0
  upper <- ceiling((n+1)*p);   lower <- floor((n+1)*p)
  while(plev < level) { upper<-upper+1; lower<-lower-1
      plev <- pbinom(upper-1, size=n, prob=p)
                                - pbinom(lower-1, size=n, prob=p) }
  x <- sort(data)
cat(100*level,"%-KI zum",p,"-Quantil aus den Daten:",
                              x[lower],"-",x[upper],"\n")
}

daten <- c(95, 84, 105, 96, 86, 95, 94, 75, 93)
ci.perc(daten, 0.5, 0.95)
## 95 %-KI zum 0.5 -Quantil aus den Daten: 84 - 96
```

H-D Schätzer nach Harrell und Davis:

Funktion hdquantile() in library(Hmisc)

```
library(Hmisc)

daten <- c(95, 84, 105, 96, 86, 95, 94, 75, 93)
quantile(daten)
##   0%  25%  50%  75% 100%
##   75   86   94   95  105

hdquantile(daten, se=TRUE)
##      0.00      0.25      0.50      0.75      1.00
##  75.00000  85.63000  92.95571  96.61704 105.00000
## attr(,"se")
##     0.00     0.25     0.50     0.75     1.00
##       NA 4.439162 2.313456 1.884757       NA

hd<-function(x, q=.5)   {                        # Harrell-Davis Schätzer für Quantile
    if(length(x)!=length(x[!is.na(x)]))
      stop("Remove missing values from x")
    n   <- length(x);     m1  <- (n+1)*q;     m2  <- (n+1)*(1-q)
    vec <- seq(along=x)
    w   <- pbeta(vec/n, m1, m2) - pbeta((vec-1)/n, m1, m2)
    y   <- sort(x)
    sum(w*y)
}

hd(daten, q=0.25)
## [1] 85.63
```

6.10.3 90%-Konfidenzintervall für Referenzwerte

Faktor nach H.E. Solberg für den parametrischen Fall:

```
alpha <- 0.05                                                    # 95% Referenbereich
beta  <- 0.10                                                    # 90% - KI

za  <- qnorm(1-alpha/2)                                          # zweiseitig
phi <- dnorm(za)
zb  <- qnorm(1-beta/2)

const <- zb*sqrt((2+za^2)/2); const                              # Solberg - Faktor
## [1] 2.811078
```

```
delta <- c(0.040, 0.030, 0.025, 0.020, 0.015, 0.01)
par    <- round((1 + 0.5*za^2)*(phi*zb/(delta/2))^2, 0)
nonpar <- round(alpha/2 * (1-alpha/2) * (zb/(delta/2))^2, 0)
tab    <- cbind(delta, par, nonpar)
rownames(tab) <- c("2.00%","1.50%","1.25%","1.00%","0.75%","0.50%")
colnames(tab) <- c("Delta","n par","n nonpar")
as.data.frame(tab)
```

	Delta	n par	n nonpar
2.00%	0.040	67	165
1.50%	0.030	120	293
1.25%	0.025	173	422
1.00%	0.020	270	659
0.75%	0.015	480	1172
0.50%	0.010	1080	2638

```
nreference <- function(alpha=0.05, side="zweiseitig", beta=0.10, delta=0.01) {
  if (side=="einseitig") {
    za <- qnorm(1-alpha); phi <- dnorm(za); zb <- qnorm(1-beta/2);
    par    <- round((1 + 0.5*za^2)*(phi*zb/delta)^2, 0)
    nonpar <- round(alpha * (1-alpha) * (zb/delta)^2, 0)  }
  if (side=="zweiseitig") {
    delta <- delta/2
    za <- qnorm(1-alpha/2); phi <- dnorm(za); zb <- qnorm(1-beta/2);
    par    <- round((1 + 0.5*za^2)*(phi*zb/delta)^2, 0)
    nonpar <- round(alpha/2 * (1-alpha/2) * (zb/delta)^2, 0)  }
  pa  <- (1-alpha)*100;   pb  <- (1-beta)*100
  cat("\n","Fallzahlabschätzung für den",pa,"%-Referenzbereich",side,"\n",
      "mit einem", pb,"%-KI und einer Toleranz von ", round(delta*100, 1), "% \n",
      "n - parametrisch      =", par, "\n","n - nichtparametrisch =", nonpar,"\n")
}

nreference(alpha=0.05, side="einseitig", beta=0.10, delta=0.01)
##
##  Fallzahlabschätzung für den 95 %-Referenzbereich einseitig
##  mit einem 90 %-KI und einer Toleranz von  1 %
##  n - parametrisch      = 677
```

```
##  n - nichtparametrisch = 1285

nreference(alpha=0.05, side="zweiseitig", beta=0.10, delta=0.01)
##
##  Fallzahlabschätzung für den 95 %-Referenzbereich zweiseitig
##  mit einem 90 %-KI und einer Toleranz von  0.5 %
##  n - parametrisch      = 1080
##  n - nichtparametrisch = 2638
```

6.11 Konfidenzintervalle nach dem Bootstrap-Verfahren

```
x <- c(68, 69, 69, 70, 71, 72, 72, 74); mean(x)
## [1] 70.625
b1 <- sample(x, 8, replace = TRUE); b1; mean(b1)
## [1] 72 68 74 72 68 72 69 74
## [1] 71.125
b2 <- sample(x, 8, replace = TRUE); b2; mean(b2)
## [1] 74 72 70 72 68 72 72 74
## [1] 71.75
b3 <- sample(x, 8, replace = TRUE); b3; mean(b3)
## [1] 69 72 71 74 72 69 72 72
## [1] 71.375
b4 <- sample(x, 8, replace = TRUE); b4; mean(b4)
## [1] 69 72 71 71 74 69 69 69
## [1] 70.5
b5 <- sample(x, 8, replace = TRUE); b5; mean(b5)
## [1] 69 70 69 69 72 72 70 70
## [1] 70.125
sd(c(mean(b1), mean(b2), mean(b3), mean(b4), mean(b5)))
## [1] 0.6578849

x <- c(68, 69, 69, 70, 71, 72, 72, 74)
b <- rep(NA, 1000)
for (i in 1:1000) b[i] <- mean(sample(x, 8, replace=TRUE))
quantile(b, probs = c(0.025, 0.975))
##   2.5%  97.5%
## 69.375 71.875
```

Bootstrap Perzentilmethode:

Funktion bootstrap() in library(bootstrap)

```
library(bootstrap)
x <- c(10, 10, 11, 12, 12, 13, 14, 15, 15, 16, 17, 20, 21, 24, 30)
n <- length(x)
boot <- bootstrap(x, 500, median)                    # Median aus 500 Stichproben
quantile(boot$thetastar, probs=c(.025,.975))         # Quantile der Verteilung
##  2.5% 97.5%
##    12    17
```

Bootstrap t-Methode:

```
library(bootstrap)
x <- c(10, 15, 20, 16, 13, 12, 15, 21, 11, 24, 17, 14, 12, 10, 30)
boott(x, median, nbootsd=50, nboott=1000, perc=c(0.025, 0.975))
## $confpoints
##          0.025    0.975
## [1,] 12.93397 18.97054
##
## $theta
## NULL
##
## $g
## NULL
##
## $call
## boott(x = x, theta = median, nbootsd = 50, nboott = 1000, perc = c(0.025,
##     0.975))

mean(x)+qt(0.025, n-1)*sd(x)/sqrt(n)          # analytischer Ansatz - Normalverteilung
## [1] 12.87434
mean(x)+qt(0.975, n-1)*sd(x)/sqrt(n)
## [1] 19.12566
```

6.12 Konfidenzintervalle für die Varianz / Standardabweichung

Angenäherte Stichprobenumfäge nach W.C. Guenther (Tabelle):

```
sL.limit <- function(n, alpha, gamma) {
  chiq1 <- sqrt(qchisq(gamma, n-1))
  chiq2 <- sqrt(qchisq(alpha/2,n-1))
  chiq3 <- sqrt(qchisq(1-alpha/2, n-1))
  L <- chiq1*(1/chiq2 - 1/chiq3)
}

weite <- sL.limit(c(75,  12,  6), 0.10, 0.90); round(weite, 1)
## [1] 0.3 1.0 1.9
weite <- sL.limit(c(88,  15,  7), 0.10, 0.99); round(weite, 1)
## [1] 0.3 1.0 2.1
weite <- sL.limit(c(105, 16,  7), 0.05, 0.90); round(weite, 1)
## [1] 0.3 1.0 2.1
weite <- sL.limit(c(120, 19,  9), 0.05, 0.99); round(weite, 1)
## [1] 0.3 1.0 2.0


mL.limit <- function(n, alpha, gamma) {
  t.q   <- qt(1-alpha/2, n-1)
  chi.q <- qchisq(gamma, n-1)
  L     <- sqrt((4*t.q^2*chi.q)/(n*(n-1)))
}

weite <- mL.limit(c(140, 18,  7), 0.10, 0.90); round(weite,1)
```

```
## [1] 0.3 1.0 2.0
weite <- mL.limit(c(150, 22,  9), 0.10, 0.99); round(weite,1)
## [1] 0.3 1.0 2.0
weite <- mL.limit(c(190, 24,  9), 0.05, 0.90); round(weite,1)
## [1] 0.3 1.0 2.0
weite <- mL.limit(c(210, 29, 11), 0.05, 0.99); round(weite,1)
## [1] 0.3 1.0 2.0
```

Konfidenzintervalle für den Variationskoeffizienten

```
cv_ki <- function(x, conf.level=0.95) {
  alpha <- 1 - conf.level
  n <- length(x)
  V <- sd(x)/mean(x)
  q1 <- qchisq(1-alpha/2, df=n-1);  q2 <- qchisq(alpha/2, df=n-1)
  low <- round(V / sqrt((q1/(n-1)+V^2*((q1+2)/n - 1))), 4)
  upp <- round(V / sqrt((q2/(n-1)+V^2*((q2+2)/n - 1))), 4)
  cat("\n","Variationskoeffizient V=",round(V, 4),"\n",
      100*conf.level,"%-Konfidenzintervall:",low,"to",upp,"\n")
}

x <- c(9.74,  9.44, 10.30,  9.39,  9.72,  9.78, 10.46,  8.98, 10.77,  9.84,
       8.77, 10.40, 10.69, 10.16, 10.36,  9.69, 10.33,  9.92,  9.82,  9.43)
cv_ki(x, conf.level=0.95)
##
##  Variationskoeffizient V= 0.0544
##  95 %-Konfidenzintervall: 0.0413 to 0.0796
```

Stichprobenumfänge zur Schätzung des Variationskoeffizienten:

Funktion ss.aipe.cv() in library(MBESS)

```
library(MBESS)                                                  # function ss.aipe()

# weite <- seq(0.02, 0.10, 0.01)
# n <- length(weite)
# size3  <- rep(NA, n)
# size5  <- rep(NA, n)
# size7  <- rep(NA, n)
# size10 <- rep(NA, n)
#
# for (i in 1:n) {
#   size3[i]  <- ss.aipe.cv(C.of.V = 0.03, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.95)
#   size5[i]  <- ss.aipe.cv(C.of.V = 0.05, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.95)
#   size7[i]  <- ss.aipe.cv(C.of.V = 0.07, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.95)
#   size10[i] <- ss.aipe.cv(C.of.V = 0.10, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.95)
```

```
# }
#
# tab1 <- rbind("3%"=ceiling(size3), "5%"=ceiling(size5),
#               "7%"=ceiling(size7), "10%"=ceiling(size10))
# colnames(tab1) <- seq(0.02, 0.10, 0.01)
# as.data.frame(tab1)

##     0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
## 3%    24   14   10    7    6    6    5    5   5
## 5%    55   28   18   14   10    9    8    7   6
## 7%   102   49   30   22   16   13   11    9   8
## 10%  203   94   56   38   27   21   17   15  13

# weite <- seq(0.02, 0.10, 0.01)
# n <- length(weite)
# size3  <- rep(NA, n)
# size5  <- rep(NA, n)
# size7  <- rep(NA, n)
# size10 <- rep(NA, n)

# for (i in 1:n) {
#   size3[i]  <- ss.aipe.cv(C.of.V = 0.03, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.99)
#   size5[i]  <- ss.aipe.cv(C.of.V = 0.05, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.99)
#   size7[i]  <- ss.aipe.cv(C.of.V = 0.07, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.99)
#   size10[i] <- ss.aipe.cv(C.of.V = 0.10, sup.int.warns=TRUE,
#                           width = weite[i], conf.level = 0.99)
#}

# tab2 <- rbind("3%"=ceiling(size3), "5%"=ceiling(size5),
#               "7%"=ceiling(size7), "10%"=ceiling(size10))
# colnames(tab2) <- seq(0.02, 0.10, 0.01)
# as.data.frame(tab2)

##     0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1
## 3%    39   22   16   13   11    8    7    7   6
## 5%    93   46   30   22   18   13   11   10   9
## 7%   174   82   50   35   27   20   17   14  13
## 10%  348  160   94   64   47   37   28   23  20
```

6.13 Weibull-Verteilung

Beispiel Garnqualität:

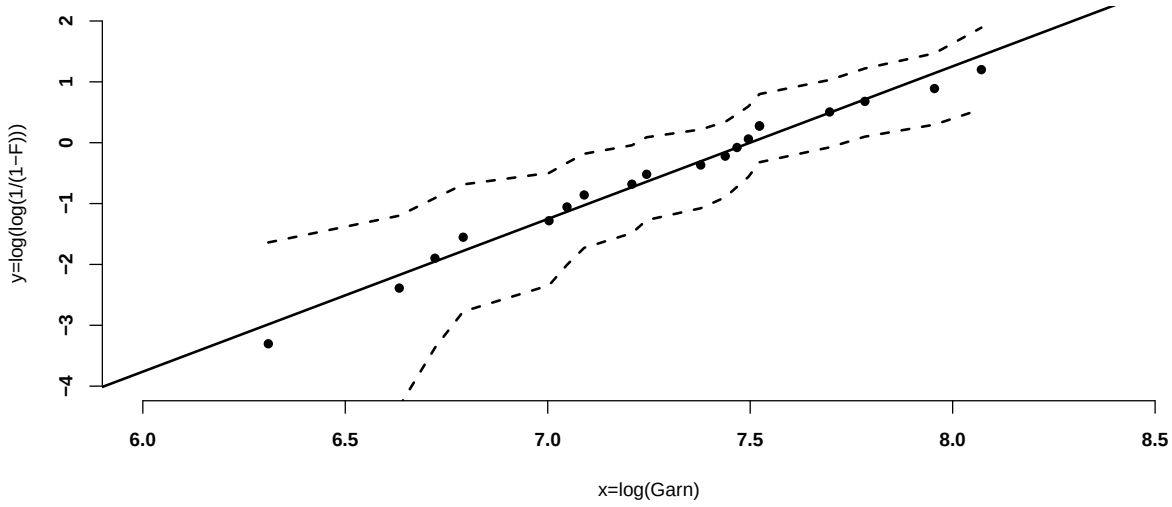
```
garn  <- c(550, 760, 830, 890, 1100, 1150, 1200, 1350, 1400, 1600,
           1700, 1750, 1800, 1850, 1850, 2200, 2400, 2850, 3200)
garn <- sort(garn);  n    <- length(garn)

F    <- (rank(garn) - 0.3) / (n+0.4)                    # empirische Verteilungsfunktion
x    <- log(garn)                                              # Transformation
y    <- log(log(1/(1-F)))
z <- lm(y ~ x); z                                              # lineare Regression
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##     -18.813        2.509
coef(z)[2]                                                     # shape
##        x
## 2.508568
exp(-(coef(z)[1]/coef(z)[2]))                                  # scale
## (Intercept)
##    1807.446
```

Funktion mle2() in library(bbmle)

```
library(bbmle)                                                 # MLE-Weibullverteilung
ll <- function(shape=1.5, scale=2000)
        - sum(stats::dweibull(garn, shape, scale, log = TRUE))

mle2(ll)
##
## Call:
## mle2(minuslogl = ll)
##
## Coefficients:
##       shape       scale
##    2.549478 1893.727998
##
## Log-likelihood: -150.41
##
## Warning: optimization did not converge (code 1: )
```

6.13.2 Konfidenzintervall für die Weibull-Gerade

```
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="n", ps=11)
plot(x, y, xlab="x=log(Garn)", ylab="y=log(log(1/(1-F)))",
     xlim=c(6,8.5), ylim=c(-4, 2), pch=16, cex=1.0)
abline(z)
i <- rank(garn);  n <-length(garn)
v.unten <- 1 / (((n-i+1)/i)*(qf(0.025, 2*(n-i+1), 2*i))+1)
v.oben  <- 1 - 1 / (1 + (i / (n-i+1)*qf(0.025, 2*i, 2*(n-i+1))))
lines(x, log(log(1/(1-v.oben))), lty=2)
lines(x, log(log(1/(1-v.unten))), lty=2)
```



6.14 Konfidenzintervalle für die Parameter einer linearen Regression

6.14.1 Schätzung einiger Standardabweichungen

```
n   <- 7
x   <- c(13, 17, 10, 17, 20, 11, 15); sum(x); sum(x^2)
## [1] 103
## [1] 1593
y   <- c(12, 17, 11, 13, 16, 14, 15); sum(y); sum(y^2)
## [1] 98
## [1] 1400
xy  <- x * y;                         sum(xy)
## [1] 1475
Qx  <- sum(x^2) - sum(x)^2/n;         Qx
## [1] 77.42857
Qy  <- sum(y^2) - sum(y)^2/n;         Qy
## [1] 28
Qxy <- sum(xy)  - sum(x)*sum(y)/n;    Qxy
## [1] 33

r   <- Qxy / sqrt(Qx*Qy);             r
## [1] 0.7087357

sx   <- sqrt(Qx/(n-1));               sx
## [1] 3.59232
sy   <- sqrt(Qy/(n-1));               sy
## [1] 2.160247
sy.x <- sqrt((Qy - Qxy^2/Qx)/(n-2));  sy.x
## [1] 1.669456

byx  <- Qxy/Qx;                        byx
## [1] 0.4261993
sbyx <- sy.x/sqrt(Qx);                 sbyx
## [1] 0.189725

ayx  <- mean(y) - byx*mean(x);         ayx
## [1] 7.728782
sayx <- sy.x*sqrt(1/n + mean(x)^2/Qx); sayx
## [1] 2.86209
```

```
summary(lm(y~x))
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       1       2       3       4       5       6       7
## -1.2694  2.0258 -0.9908 -1.9742 -0.2528  1.5830  0.8782
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   7.7288     2.8621   2.700   0.0428 *
## x             0.4262     0.1897   2.246   0.0746 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.669 on 5 degrees of freedom
## Multiple R-squared:  0.5023, Adjusted R-squared:  0.4028
## F-statistic: 5.046 on 1 and 5 DF,  p-value: 0.07461
```

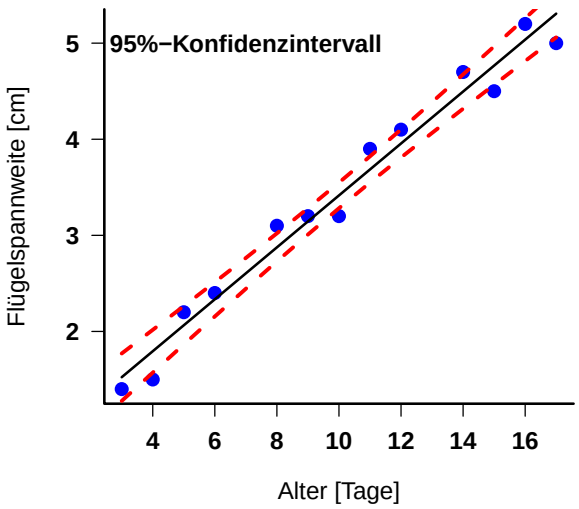
6.14.3 Prädiktionsintervalle

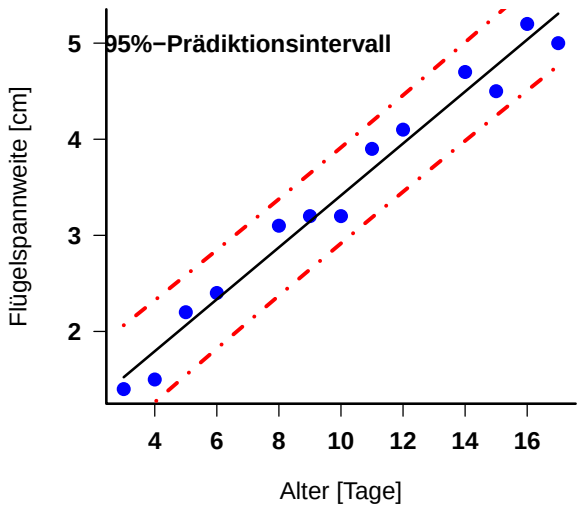
```
new <- data.frame(x = c(12,14,16,18))

predict(lm(y~x), new, int="c", level = 0.95)
##        fit      lwr      upr
## 1 12.84317 10.74953 14.93681
## 2 13.69557 12.03656 15.35458
## 3 14.54797 12.80896 16.28698
## 4 15.40037 13.12028 17.68046

predict(lm(y~x), new, int="p", level = 0.95)
##        fit       lwr      upr
## 1 12.84317  8.068231 17.61812
## 2 13.69557  9.094586 18.29656
## 3 14.54797  9.917538 19.17840
## 4 15.40037 10.540783 20.25996
```

Konfidenz- und Prädiktionsintervall am Beispiel Flügelweite von Sperlingen:

```
alter   <- c(  3,  4,  5,  6,  8,  9, 10, 11, 12, 14, 15, 16, 17)   # Tage
fluegel <- c(1.4,1.5,2.2,2.4,3.1,3.2,3.2,3.9,4.1,4.7,4.5,5.2,5.0)   # cm
labx <- "Alter [Tage]"; laby <- "Flügelspannweite [cm]"

mod <- lm(fluegel ~ alter)                                          # lineares Modell in R
summary(mod)
##
## Call:
## lm(formula = fluegel ~ alter)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.30699 -0.21538  0.06553  0.16324  0.22507
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.71309    0.14790   4.821 0.000535 ***
## alter        0.27023    0.01349  20.027 5.27e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2184 on 11 degrees of freedom
```

```
## Multiple R-squared:  0.9733, Adjusted R-squared:  0.9709
## F-statistic: 401.1 on 1 and 11 DF,  p-value: 5.267e-10

a <- mod$coef[1]; round(a, 2)                                    # Achsenabschnitt
## (Intercept)
##        0.71

b <- mod$coef[2]; round(b, 3)                                    # Regressionskoeffizient
## alter
##  0.27

par(mfrow=c(1,2), lwd=2, font=2, font.axis=2, bty="l", ps=14)
plot (alter, fluegel, type="p", las=1,
      xlab=labx, ylab=laby, cex=1.5, pch=16, col = "blue")
text(7, 5, "95%-Konfidenzintervall")
chol.est <- a + b * alter
lines(alter, chol.est, lty=1.2, col="black")

newx  <- seq(3, 17, by=1)
conf_band <- predict(mod, newdata=data.frame(alter=newx),        # Konfidenzintervall
                     interval="confidence", level = 0.95)
lines(newx, conf_band[,2], col="red", lty=2, lwd=3)
lines(newx, conf_band[,3], col="red", lty=2, lwd=3)

plot (alter, fluegel, type="p", las=1,
      xlab=labx, ylab=laby, cex=1.5, pch=16, col = "blue")
text(7, 5, "95%-Prädiktionsintervall")
chol.est <- a + b * alter
lines(alter, chol.est, lty=1.2, col="black")

conf_band <- predict(mod, newdata=data.frame(alter=newx),        # Prädiktionsintervall
                     interval="prediction", level = 0.95)
lines(newx, conf_band[,2], col="red", lty=4, lwd=3)
lines(newx, conf_band[,3], col="red", lty=4, lwd=3)
```


95%−Konfidenzintervall
Flügelspannweite [cm]
Alter [Tage]
2
3
4
5
4
6
8
10
12
14
16


95%−Prädiktionsintervall
Flügelspannweite [cm]
Alter [Tage]
2
3
4
5
4
6
8
10
12
14
16

6.15 Konfidenzintervall für den Korrelationskoeffizienten

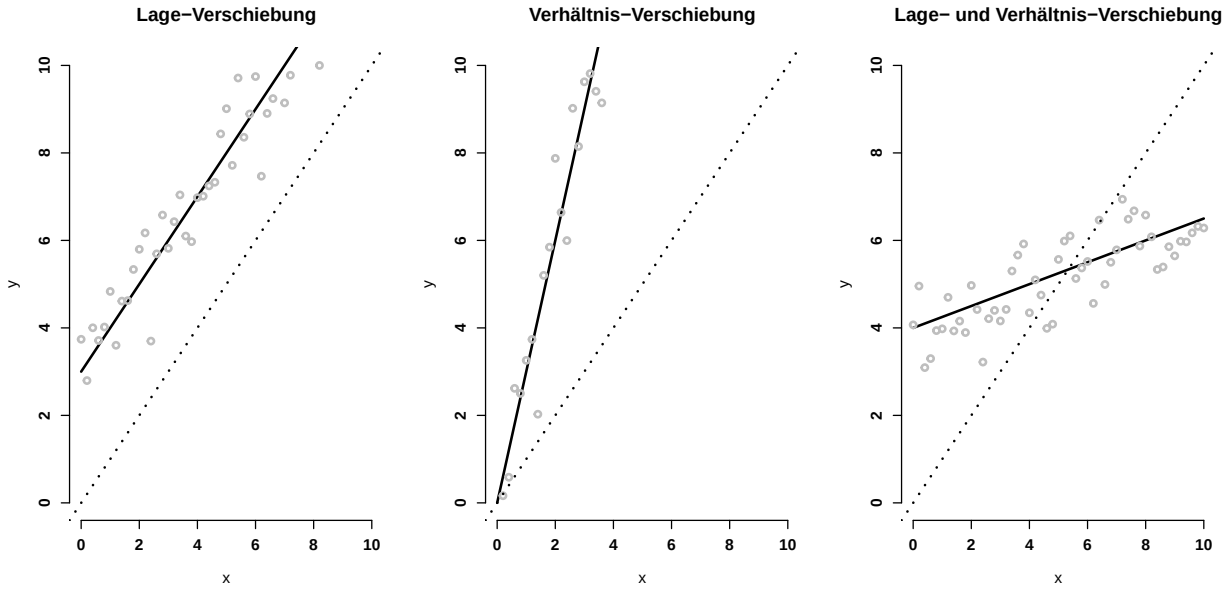
```
n <- 50
r <- 0.687

zp <- 0.5*log((1+r)/(1-r)); zp                                    # Fisher-Transformation
## [1] 0.8422519
szp <- 1/sqrt(n-3)
lwr.z <- zp - qnorm(0.975)*szp; upr.z <- zp + qnorm(0.975)*szp
lwr.z; upr.z
## [1] 0.5563618
## [1] 1.128142

lwr.r <- (exp(2*lwr.z)-1)/(exp(2*lwr.z)+1)
upr.r <- (exp(2*upr.z)-1)/(exp(2*upr.z)+1)
lwr.r; upr.r
## [1] 0.5052731
## [1] 0.8103824

                                                  # Stichprobenumfang zur Schätzung von Rho
z.trans <-  function(r) 0.5*log((1+r)/(1-r))
r.u = 0.50; r.o <- 0.80; alpha <- 0.05
n <- 4*(qnorm(1-alpha/2)/(z.trans(r.o)-z.trans(r.u)))^2 + 3
n
## [1] 53.92456
```

Stichprobenumfang zur Schätzung des Korrelationskoeffizienten:

```
z.trans <-  function(r) 0.5*log((1+r)/(1-r))
r.u <- 0.50
r.o <- 0.80
alpha <- 0.05
n <- 4*(qnorm(1-alpha/2)/(z.trans(r.o)-z.trans(r.u)))^2 + 3

n
## [1] 53.92456
```

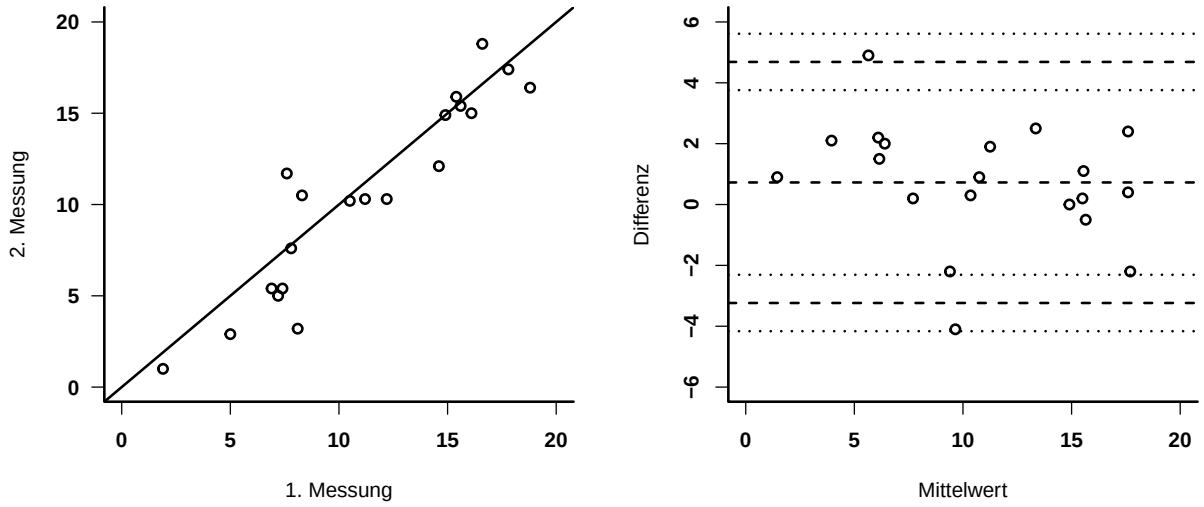
6.16 Übereinstimmung und Präzision von Messwerten

```
x  <- seq(0, 10, by=0.2); n <- length(x)
y  <- x
y1 <- x+3;          y1v <- y1 + rnorm(n, 0, 0.7)
y2 <- 3*x;          y2v <- y2 + rnorm(n, 0, 1)
y3 <- 0.25*x + 4;   y3v <- y3 + rnorm(n, 0, 0.5)

par(mfrow=c(1,3), lwd=2, font.axis=2, bty="n", ps=14)
plot(x, y1, xlab="x", ylab="y", main="Lage-Verschiebung",
        type="l", xlim=c(0,10), ylim=c(0,10))
points(x, y1v, col="grey"); abline(0, 1, lty=3)
plot(x, y2, xlab="x", ylab="y", main="Verhältnis-Verschiebung",
        type="l", xlim=c(0,10), ylim=c(0,10))
points(x, y2v, col="grey"); abline(0, 1, lty=3)
plot(x, y3, xlab="x", ylab="y", main="Lage- und Verhältnis-Verschiebung",
        type="l", xlim=c(0,10), ylim=c(0,10))
points(x, y3v, col="grey"); abline(0, 1, lty=3)
```



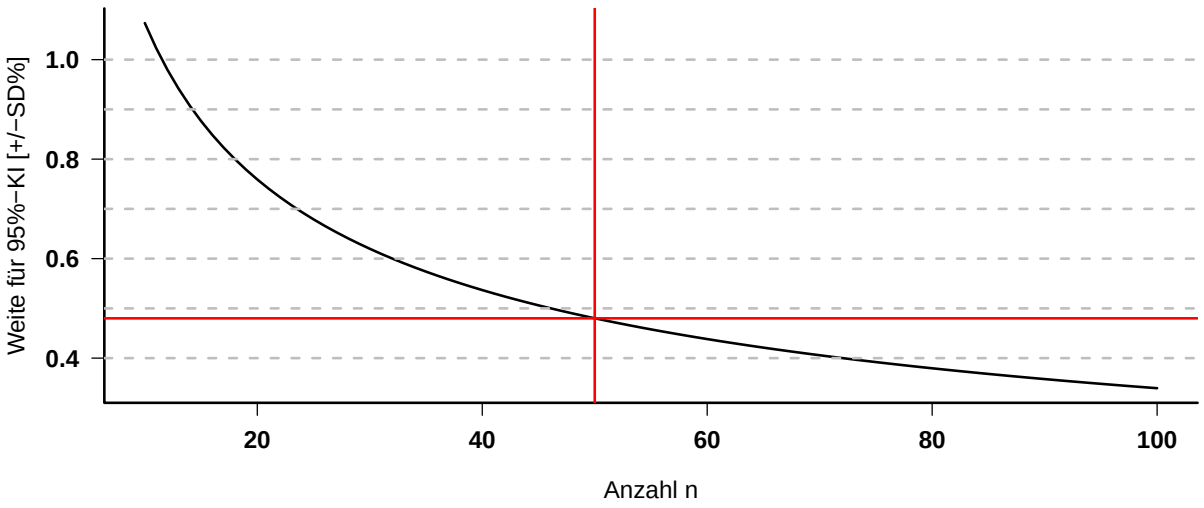
6.16.1 Übereinstimmung nach Bland Altman

```
x1 <- c(16.6, 17.8,  6.9,  7.6,  7.8,  7.4, 11.2, 16.1, 14.6,  5.0, 18.8,
        10.5,  8.1, 15.4,  7.2, 12.2,  1.9, 15.6, 14.9,  8.3)
x2 <- c(18.8, 17.4,  5.4, 11.7,  7.6,  5.4, 10.3, 15.0, 12.1,  2.9, 16.4,
        10.2,  3.2, 15.9,  5.0, 10.3,  1.0, 15.4, 14.9, 10.5)

cor(x1, x2)
## [1] 0.9290261
diff     <-  x1 - x2;     mittel <- (x1 + x2)/2
mdiff    <- mean(diff); mdiff
## [1] 0.725
sdiff    <- sd(diff);   sdiff
## [1] 1.980397
upplim   <- mdiff + 2*sdiff; upplim
## [1] 4.685795
lowlim   <- mdiff - 2*sdiff; lowlim
## [1] -3.235795

n        <- length(diff)
tval     <- qt(0.025, n-1, lower.tail=F)
upp95    <- mdiff + tval * sqrt(sdiff^2/n); upp95
## [1] 1.651854
low95    <- mdiff - tval * sqrt(sdiff^2/n); low95
## [1] -0.2018545

upp95u   <- upplim + tval * sqrt(sdiff^2/n); upp95u
## [1] 5.612649
upp95l   <- upplim - tval * sqrt(sdiff^2/n); upp95l
## [1] 3.75894

low95u  <- lowlim + tval * sqrt(sdiff^2/n); low95u
## [1] -2.30894
low95l  <- lowlim - tval * sqrt(sdiff^2/n); low95l
## [1] -4.162649

par(lwd=2, font.axis=2, bty="l", ps=12, mfrow=c(1,2))
plot(x1, x2, xlim=c(0,20), ylim=c(0,20), cex=1.0, las=1,
     xlab="1. Messung", ylab="2. Messung")
abline(0,1,col="black")
plot(mittel, diff, xlim=c(0,20), cex=1.0,
     ylim=c(-6, +6),xlab="Mittelwert", ylab="Differenz")
abline(h=mdiff,  col="black", lty=2)
abline(h=upplim, col="black", lty=2); abline(h=lowlim, col="black", lty=2)
abline(h=upp95u, col="black", lty=3); abline(h=upp95l, col="black", lty=3)
abline(h=low95u, col="black", lty=3); abline(h=low95l, col="black", lty=3)
```



Fallzahlabschätzung:

```
n  <- 10:100; CI <- 1.96*sqrt(3/n)                    # 95%-KI: +/-1.96*sqrt(3/n)*SD
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14)
plot(n, CI, type="l", xlab="Anzahl n",
     ylab="Weite für 95%-KI [+/-SD%]", las=1)
abline(h=seq(0.4,1.0,0.1), lty=2, col="grey")
abline(v=50, col="red")
abline(h=0.48, col="red")
```



6.16.2 Regressionsverfahren zur Übereinstimmung

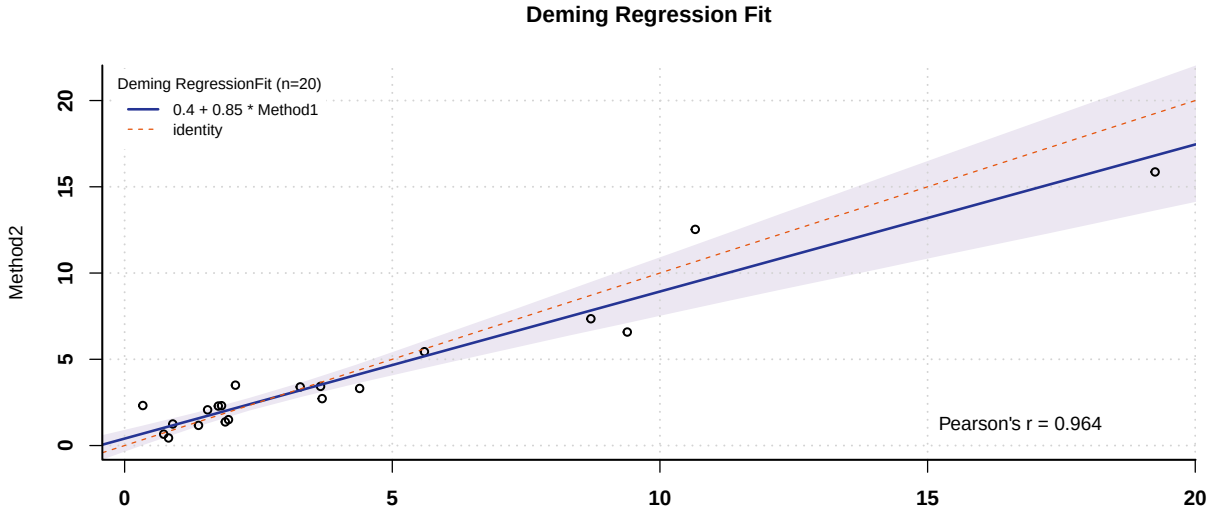
Deming Regression:

```
x <- c(8.71, 3.28, 5.60, 1.55, 1.75, 0.73, 3.66, 0.90, 9.39, 4.39,
       3.69, 0.34, 1.94, 2.07, 1.38, 1.81, 0.82, 1.88,10.66,19.25)
y <- c(7.35, 3.40, 5.44, 2.07, 2.29, 0.66, 3.43, 1.25, 6.58, 3.31,
       2.72, 2.32, 1.50, 3.50, 1.17, 2.31, 0.44, 1.37,12.53,15.86)

fitx <- lsfit(y, x); fitx$coefficients                          # lineare Regression
##  Intercept          X
## -0.2797996  1.1244779
fity <- lsfit(x, y); fity$coefficients
## Intercept         X
## 0.5114538 0.8266220

rho <- 1                                                        # Deming Regression
mx <- mean(x); my <- mean(y)
vx <- var(x);  vy <- var(y); vxy <- cov(x,y)

b1.deming <- ((rho*vy - vx) + sqrt((vx - rho*vy)^2 +
              4*rho*vxy^2))/(2*rho*vxy); b1.deming
## [1] 0.8525334
b0.deming <- my - b1.deming*mx; b0.deming
## [1] 0.4028851

di   <- y - (b0.deming + b1.deming*x)                           # estimates for x and y
xhat <- x + (rho*b1.deming*di)/(1+rho*b1.deming^2)
yhat <- y - di/(1 + rho*b1.deming^2)
```

Funktion mcreg() in library(mcr)

```
library(mcr)                                          # Deming Regression in library(mcr)
fit <- mcreg(x, y, error.ratio=rho, method.reg="Deming")
printSummary(fit)
##
##
## ------------------------------------------
##
## Reference method: Method1
## Test method:     Method2
## Number of data points: 20
##
## ------------------------------------------
##
## The confidence intervals are calculated with
##  bootstrap  ( quantile ) method.
## Confidence level: 95%
## Error ratio: 1
##
## ------------------------------------------
##
## DEMING REGRESSION FIT:
```

```
##
##                  EST SE        LCI      UCI
## Intercept 0.4028851 NA -0.3240175 0.911547
## Slope     0.8525334 NA  0.6754452 1.114265
##
##
## ----------------------------------------
##
## BOOTSTRAP SUMMARY
##
##           global.est bootstrap.mean     bias bootstrap.se
## Intercept    0.40289        0.35822 -0.04466      0.30412
## Slope        0.85253        0.86402  0.01149      0.10579
##
## Bootstrap results generated with environment RNG settings.
## NULL
par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=11, cex.axis=1.1)
plot(fit)
```



The 0.95−confidence bounds are calculated with the bootstrap(quantile) method.

```
# r   <- cor(x,y); byx <- fity$coefficients[2]
# U <- byx/(2*r^2) - (1/rho)/(2*byx)
# b1.short <- U + sqrt(U^2 + 1/rho); b1.short                # short Deming-Regression
# b1.short <- byx/r; b1.short                                # simple for rho=1
```

Passing-Bablok Regression:

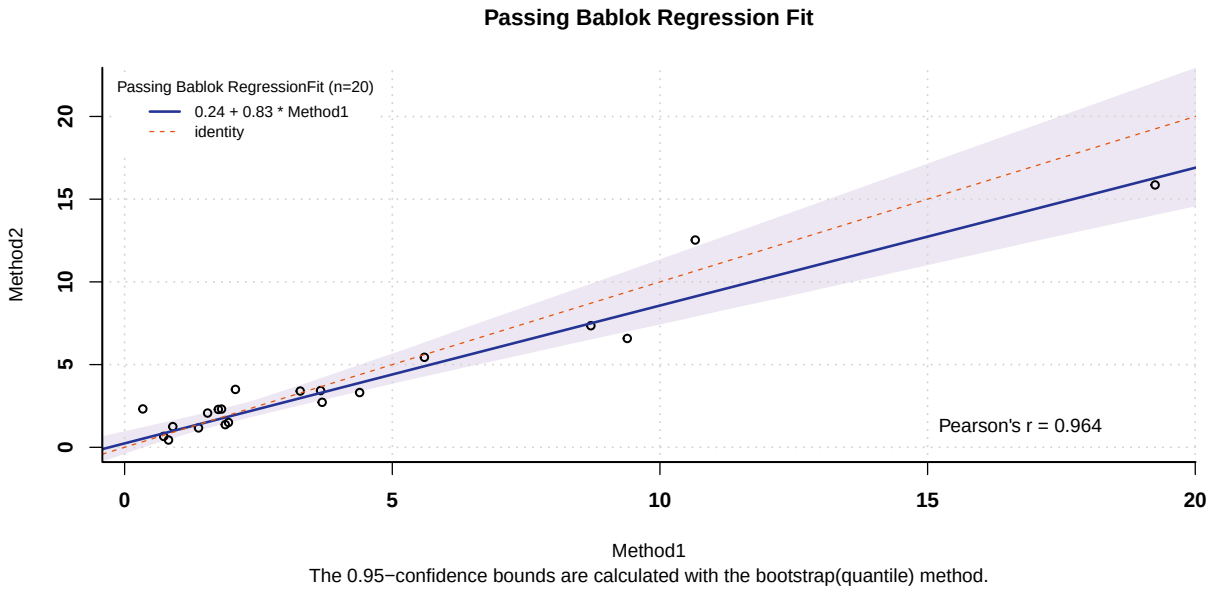
```
x <- c(8.71, 3.28, 5.60, 1.55, 1.75, 0.73, 3.66, 0.90, 9.39, 4.39,
       3.69, 0.34, 1.94, 2.07, 1.38, 1.81, 0.82, 1.88,10.66,19.25)
y <- c(7.35, 3.40, 5.44, 2.07, 2.29, 0.66, 3.43, 1.25, 6.58, 3.31,
       2.72, 2.32, 1.50, 3.50, 1.17, 2.31, 0.44, 1.37,12.53,15.86)
```

```
fitx <- lsfit(y, x); fitx$coefficients                          # lineare Regression
##  Intercept          X
## -0.2797996  1.1244779
fity <- lsfit(x, y); fity$coefficients
## Intercept         X
## 0.5114538 0.8266220

n <- length(x);   s <- array(NA, dim=c(n,n))
for(i in 1:(n-1)) {                                             # alle paarweise kombinieren
  for(j in (i+1):n) {
    if(i != j) { s[i,j] <- (y[i] - y[j])/(x[i] - x[j]) }  }  }
s <- sort(na.exclude(as.vector(s)))

K <- sum(s <= -1) - .5 * sum(s == -1); N <- length(s)           # shift median
b1.passing <- ifelse(N%%2,s[(N+1)/2+K],mean(s[N/2+K+0:1])); b1.passing
## [1] 0.8331478
b0.passing <- median(y - b1.passing*x); b0.passing
## [1] 0.2369807
```

Funktion mcreg() in library(mcr)

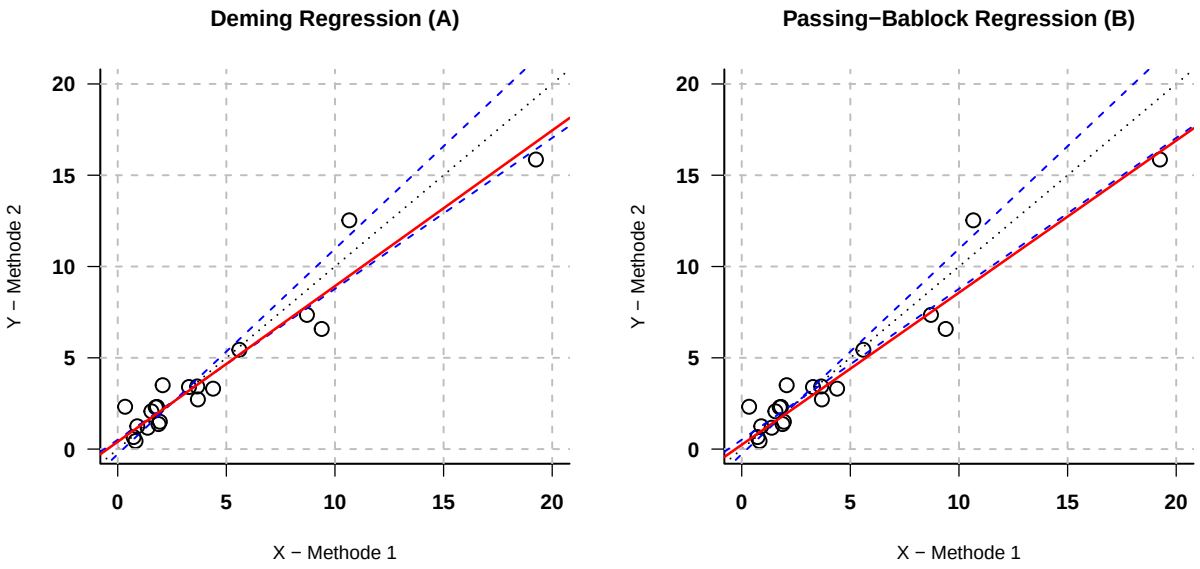
```
library(mcr)                                  # Passing-bablok Regression in library(mcr)
fit <- mcreg(x, y, method.reg="PaBa")
printSummary(fit)
##
##
## ----------------------------------------
##
## Reference method: Method1
## Test method:     Method2
## Number of data points: 20
##
## ----------------------------------------
##
## The confidence intervals are calculated with
##  bootstrap  ( quantile ) method.
## Confidence level: 95%
##
##
## ----------------------------------------
##
## PASSING BABLOK REGRESSION FIT:
##
##                EST SE        LCI       UCI
## Intercept 0.2369840 NA -0.3716544 0.9705179
## Slope     0.8331473 NA  0.6941912 1.1492705
##
##
## ----------------------------------------
##
## BOOTSTRAP SUMMARY
##
##           global.est bootstrap.mean    bias bootstrap.se
```

```
## Intercept    0.23698        0.27833 0.04135      0.36032
## Slope        0.83315        0.86228 0.02913      0.11246
##
## Bootstrap results generated with environment RNG settings.
## NULL
par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=11, cex.axis=1.1)
plot(fit)
```



The 0.95−confidence bounds are calculated with the bootstrap(quantile) method.

```
par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="l", ps=11, cex.axis=1.1)
plot(x, y, xlab="X - Methode 1", ylab="Y - Methode 2", las=1,
     xlim=c(0,20), ylim=c(0,20), main="Deming Regression (A)", cex=1.5)
abline(a = 0, b = 1, lty=3)
abline(h=seq(0,20,5), lty=2, col="grey")
abline(v=seq(0,20,5), lty=2, col="grey")
abline(fity, col="blue", lty=2); abline(fitx, col="blue", lty=2)
abline(a = b0.deming, b = b1.deming, col = "red", lty=1, lwd=2)

plot(x, y, xlab="X - Methode 1", ylab="Y - Methode 2", las=1,
     xlim=c(0,20), ylim=c(0,20), main="Passing-Bablock Regression (B)", cex=1.5)
abline(a = 0, b = 1, lty=3)
abline(h=seq(0,20,5), lty=2, col="grey")
abline(v=seq(0,20,5), lty=2, col="grey")
abline(fity, col="blue", lty=2); abline(fitx, col="blue", lty=2)
abline(a = b0.passing, b = b1.passing, col = "red", lty=1, lwd=2)
```


Deming Regression (A)
Y − Methode 2
X − Methode 1
0
5
10
15
20
Passing−Bablock Regression (B)
Y − Methode 2
X − Methode 1
0
5
10
15
20

6.16.3 Vergleich der Präzision / Genauigkeit zweier Messwertreihen

Bradley-Blackwood Test:

```
x <- c(4.80, 4.75, 4.34, 5.10, 4.47, 4.02, 4.43, 6.45, 5.36, 6.63)
y <- c(4.62, 4.73, 4.84, 4.98, 4.05, 4.35, 4.84, 5.47, 5.02, 5.99)

bradley.blackwood.test <- function(x, y) {                  # Bradley-Blackwood Test
  S <- x + y; mS <- mean(S); sdS <- sd(S)
  D <- x - y; mD <- mean(D); sdD <- sd(D)
  n <- length(D)
                                                            # lineare Regression
  mod <- lm(D~S);   RSE <- sum(mod$residuals^2)
                                                            # F-Statistik Bradley-Blackwood
  F.stat <- ((sum(D^2) - RSE)/2)/(RSE/(n-2))
  p.val  <- pf(F.stat, 2, n-2, lower.tail=F)
  cat("\n","Reihe X   - Mittelwert:",round(mean(x),3)," - Varianz:",round(var(x),4),
      "\n","Reihe Y   - Mittelwert:",round(mean(y),3)," - Varianz:",round(var(y),4),
      "\n","Differenz - Mittelwert:",round(mean(D),3)," - Varianz:",round(var(D),4),
      "\n","Bradley-Blackwood Test: F=",round(F.stat,3),
      "(p=",round(p.val,4),")","\n")
}

bradley.blackwood.test(x, y)
##
##  Reihe X   - Mittelwert: 5.035  - Varianz: 0.7768
##  Reihe Y   - Mittelwert: 4.889  - Varianz: 0.2969
##  Differenz - Mittelwert: 0.146  - Varianz: 0.2248
##  Bradley-Blackwood Test: F= 5.465 (p= 0.0319 )
```

6.16.4 Konkordanzkoeffizient nach Lin

Beispiel Beikonzentrationen:

```
lead <- data.frame(matrix(
  c(1,  0.22,   0.21,  2,   0.26,   0.23, 3,    0.30,   0.27,  4,   0.33,   0.27,
    5,  0.36,   0.31,  6,   0.39,   0.33, 7,    0.41,   0.37,  8,   0.44,   0.38,
    9,  0.47,   0.40,  10,  0.51,   0.43,  11, 0.55,  0.47),
  byrow=T, nrow = 11)); dimnames(lead)[[2]] <- c("sample","X","Y")
as.data.frame(t(lead))
```

	1	2	3	4	5	6	7	8	9	10	11
sample	1.00	2.00	3.00	4.00	5.00	6.00	7.00	8.00	9.00	10.00	11.00
X	0.22	0.26	0.30	0.33	0.36	0.39	0.41	0.44	0.47	0.51	0.55
Y	0.21	0.23	0.27	0.27	0.31	0.33	0.37	0.38	0.40	0.43	0.47

```
attach(lead)
                                                            # elementare Berechnung
rho.c  <- 2*cov(X,Y) / (var(X) + var(Y) + (mean(X) - mean(Y))^2); rho.c
## [1] 0.8446376
```

```
ciconcord <- function(x, y, alpha=0.05) {           # Konfidenzschranken
  zquant <- qnorm(1-alpha/2); n  <- length(X)
  r <- cor(X, Y, method = "pearson")
  rho.c  <- 2*cov(X,Y) / (var(X) + var(Y) + (mean(X) - mean(Y))^2)
  zc <- 0.5*log((1+rho.c)/(1-rho.c))
  u <- ((n-1)*(mean(X) - mean(Y))^2) /
      sqrt(sum((X-mean(X))^2)*sum((Y-mean(Y))^2))
  vzc <- (((1-r^2)*rho.c^2) / ((1-rho.c^2)*r^2)
          + (4*rho.c^3*(1-rho.c)*u^2) / (r*(1-rho.c^2)^2)
          - (2*rho.c^4*u^4) / (r^2*(1-rho.c^2)^2)) / (n-2)
  sezc <- sqrt(vzc)/sqrt(n)
  lwr.zc <- zc - zquant*sezc
  upr.zc <- zc + zquant*sezc
  lwr.r <- (exp(2*lwr.zc)-1)/(exp(2*lwr.zc)+1)
  upr.r <- (exp(2*upr.zc)-1)/(exp(2*upr.zc)+1)
  cat("\n", (1-alpha)*100,"%-Konfidenzintervall für den Konkordanz-","\n",
      "Korrelationskoeffizienten (", round(rho.c,3),"): [",
      round(lwr.r,3),";",round(upr.r,3),"]","\n")
}

ciconcord(lead$X, lead$Y)
##
##  95 %-Konfidenzintervall für den Konkordanz-
##  Korrelationskoeffizienten ( 0.845 ): [ 0.808 ; 0.875 ]
```

6.16.5 Intraklassen-Korrelation: Interrater-Reliabilität

Beispiel Hypophysenhöhe:

```
hypo <- as.data.frame(
        matrix(c(11.70,   12.50,   11.30,  7.10,    7.50,    7.80,
                  9.60,   10.30,    9.60,  7.60,    7.70,    7.40,
                  6.00,    5.90,    5.80,  6.40,    6.70,    6.20,
                  8.30,    8.30,    8.40,  9.60,    9.80,    9.90,
                  3.00,    3.40,    4.40,  5.20,    5.70,    5.40),
                  nrow = 10, ncol = 3, byrow = TRUE,
                  dimnames = list(1:10, c("U1", "U2", "U3"))))
hypo
```

U1	U2	U3
11.7	12.5	11.3
7.1	7.5	7.8
9.6	10.3	9.6
7.6	7.7	7.4
6.0	5.9	5.8
6.4	6.7	6.2
8.3	8.3	8.4
9.6	9.8	9.9
3.0	3.4	4.4
5.2	5.7	5.4

```
ICC <- function(x, t) {                          # Intraklassen-Korrelationskoeffizient
    # Summenbildung
    n <- dim(x)[1]; k <- dim(x)[2]
    col.sums  <- apply(x, 2, sum);   row.sums  <- apply(x, 1, sum)
    tot.sum   <- sum(col.sums);
    col2.sums <- apply(x^2, 2, sum); tot2.sum  <- sum(col2.sums)

    SSt       <- tot2.sum          - tot.sum^2/(n*k)      # Varianzkomponenten
    SSa       <- sum(row.sums^2)/k - tot.sum^2/(n*k)
    SSb       <- sum(col.sums^2)/n - tot.sum^2/(n*k)
    SSe       <- SSt - SSa - SSb
                                                         # ANOVA nach Shrout - Fleiss
    BMS       <- SSa/(n-1);       WMS       <- (SSt-SSa)/(n*(k-1))
    JMS       <- SSb/(k-1);       EMS       <- SSe/((n-1)*(k-1))

    if (t==1) {
        ICC <- (BMS-WMS)/(BMS+(k-1)*WMS)
        Qms <- BMS / WMS
        quf <- qf(0.975, n-1, n*(k-1)); qof <- qf(0.025, n-1, n*(k-1))
        liu <- (Qms - quf)/(Qms + (k-1)*quf)
        lio <- (Qms - qof)/(Qms + (k-1)*qof) }
    if (t==2) {
        ICC <- (BMS-EMS)/(BMS+(k-1)*EMS+(k*JMS-EMS)/n)
        Fj  <- JMS/EMS
        nue <- ((k-1)*(n-1)*(k*ICC*Fj+n*(1+(k-1)*ICC)-k*ICC)^2) /
            ((n-1)*k^2*ICC^2*Fj^2+(n*(1+(k-1)*ICC)-k*ICC)^2)
        quf <- qf(0.975, n-1, nue); qof <- qf(0.975, nue, n-1)
        liu <- (n*(BMS-quf*EMS))/(quf*(k*JMS+(k*n-k-n)*EMS)+n*BMS)
        lio <- (n*(qof*BMS-EMS))/(k*JMS+(k*n-k-n)*EMS+n*qof*BMS) }
    if (t==1 | t==2) {
        cat("ICC Typ(",t,") = ", round(ICC,4), "\n", "95%-Konfidenzintervall: ",
            round(liu,4)," - ",round(lio,4),"\n") }
    else cat("Typ unbekannt","\n")
}

ICC(hypo, 2)
## ICC Typ( 2 ) =  0.9759
##  95%-Konfidenzintervall:  0.9353  -  0.9938
```

Funktion icc() in library(irr)

```
library(irr)
icc(hypo, "oneway", "agreement")                 # Funktion icc() in library(irr)
##  Single Score Intraclass Correlation
##
##    Model: oneway
##    Type : agreement
##
##    Subjects = 10
##      Raters = 3
##      ICC(1) = 0.977
##
```

```
##  F-Test, H0: r0 = 0 ; H1: r0 > 0
##     F(9,20) = 130 , p = 9.67e-16
##
##  95%-Confidence Interval for ICC Population Values:
##   0.937 < ICC < 0.994
```

6.17 Toleranzgrenzen

```
tol_int <- function(gamma, n, P=0.95, dir="zweiseitig") {
  if (dir=="einseitig")
    k <- qt(gamma, n-1, qnorm(P)*sqrt(n))/sqrt(n)
  if (dir=="zweiseitig")
    k <- sqrt(((n-1)*(1+1/n)*(qnorm((1-P)/2)^2))/qchisq(1-gamma,n-1))
  cat("Der Toleranzfaktor (",dir,") f?r n =",n,"\n",
      "Beboachtungen mit Gamma =",gamma,"und P =",P,"ist k =",k,"\n") }

tol_int(0.80, n=10, P=0.90, dir="einseitig")
## Der Toleranzfaktor ( einseitig ) f?r n = 10
##  Beboachtungen mit Gamma = 0.8 und P = 0.9 ist k = 1.770137
```

Tabelle:

```
n <- c(seq(5, 50, 5), seq(60, 100, 10)); nl <- length(n)

tabn  <- c("Anzahl n", n)
tab1  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))
tab2  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))
tab3  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))
tab4  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))
tab5  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))
tab6  <- as.data.frame(matrix(rep(NA, 2*(nl+1)), ncol=2, nrow=nl+1))

gamma <- 0.90; p <- 0.95
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab1[1,1] <- "k einseitig";    tab1[1:nl+1, 1] <- round(k1, 2)
tab1[1,2] <- "k zweiseitig ";  tab1[1:nl+1, 2] <- round(k2, 2)

gamma <- 0.95; p <- 0.95
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab2[1,1] <- "k einseitig";    tab2[1:nl+1, 1] <- round(k1, 2)
tab2[1,2] <- "k zweiseitig ";  tab2[1:nl+1, 2] <- round(k2, 2)

gamma <- 0.99; p <- 0.95
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab3[1,1] <- "k einseitig";    tab3[1:nl+1, 1] <- round(k1, 2)
tab3[1,2] <- "k zweiseitig ";  tab3[1:nl+1, 2] <- round(k2, 2)
```

```
gamma <- 0.90; p <- 0.99
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab4[1,1] <- "k einseitig";    tab4[1:nl+1, 1] <- round(k1, 2)
tab4[1,2] <- "k zweiseitig ";  tab4[1:nl+1, 2] <- round(k2, 2)

gamma <- 0.95; p <- 0.99
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab5[1,1] <- "k einseitig";    tab5[1:nl+1, 1] <- round(k1, 2)
tab5[1,2] <- "k zweiseitig ";  tab5[1:nl+1, 2] <- round(k2, 2)

gamma <- 0.99; p <- 0.99
k1 <- qt(gamma, n-1, qnorm(p)*sqrt(n))/sqrt(n)
k2 <- sqrt(((n-1)*(1+1/n)*(qnorm((1-p)/2)^2))/qchisq(1-gamma,n-1))
tab6[1,1] <- "k einseitig";    tab6[1:nl+1, 1] <- round(k1, 2)
tab6[1,2] <- "k zweiseitig ";  tab6[1:nl+1, 2] <- round(k2, 2)

tab <- cbind(tabn, tab1, tab2, tab3, tab4, tab5, tab6)
as.data.frame(tab[,1:8])
```

tabn	V1	V2	V1.1	V2.1	V1.2	V2.2	V1.3
Anzahl n	k einseitig	k zweiseitig	k einseitig	k zweiseitig	k einseitig	k zweiseitig	k einseitig
5	3.4	4.16	4.2	5.09	6.58	7.88	4.67
10	2.57	3.02	2.91	3.38	3.74	4.27	3.53
15	2.33	2.71	2.57	2.95	3.1	3.51	3.21
20	2.21	2.56	2.4	2.75	2.81	3.17	3.05
25	2.13	2.47	2.29	2.63	2.63	2.97	2.95
30	2.08	2.41	2.22	2.55	2.52	2.84	2.88
35	2.04	2.37	2.17	2.49	2.43	2.75	2.83
40	2.01	2.33	2.13	2.44	2.36	2.68	2.79
45	1.99	2.31	2.09	2.41	2.31	2.62	2.76
50	1.97	2.28	2.06	2.38	2.27	2.58	2.73
60	1.93	2.25	2.02	2.33	2.2	2.51	2.69
70	1.91	2.22	1.99	2.3	2.15	2.45	2.66
80	1.89	2.2	1.96	2.27	2.11	2.41	2.64
90	1.87	2.19	1.94	2.25	2.08	2.38	2.62
100	1.86	2.17	1.93	2.23	2.06	2.36	2.6

6.18 Voraussageintervalle

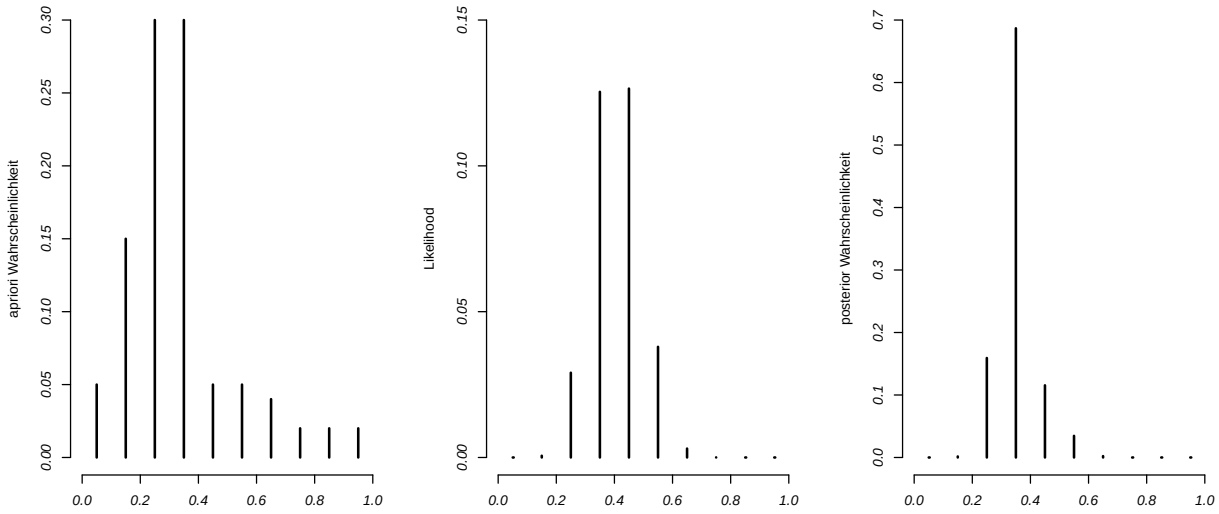
```
n <- 5; xbar <- 50.10; stdabw <- 1.31; m <- 3
xbar - qt(0.975, n-1)*stdabw*sqrt(1/m+1/n)
## [1] 47.44381
xbar + qt(0.975, n-1)*stdabw*sqrt(1/m+1/n)
## [1] 52.75619
```

6.19 Bayes-Schätzung

Beispiel Langschläfer:

```
p     <- seq(0.05, 0.95, by=0.1)                        #  diskrete a-priori Verteilung
prior <- c (5, 15, 30, 30, 5, 5, 4, 2, 2, 2);   nl <- length(prior)
prior <- prior/sum(prior)

success <- 12; failure <- 18                                    # Stichprobe
n       <- success + failure;   p_hat   <- success / n

Likel   <- rep(NA, nl)                                          # Likelihood
for (i in 0:nl) Likel[i] <- choose(n, success)*p[i]**success *
            (1-p[i])**(n-success)

posterior <- (Likel * prior) / sum(Likel*prior)                 # posterior Verteilung
```

```
par(mfrow=c(1,3), lwd=2, bty="n", font=1, font.axis=3, font.lab=1, ps=12)
plot(p, prior, type="h", ylab="apriori Wahrscheinlichkeit", xlab=" ",
                         ylim=c(0, 0.30), xlim=c(0, 1))
plot(p, Likel, type="h", ylab="Likelihood", xlab=" ",
                 ylim=c(0, 0.15), xlim=c(0, 1))
plot(p, posterior, type="h", ylab="posterior Wahrscheinlichkeit", xlab=" ",
                         ylim=c(0, 0.7), xlim=c(0, 1))
```



Tabelle

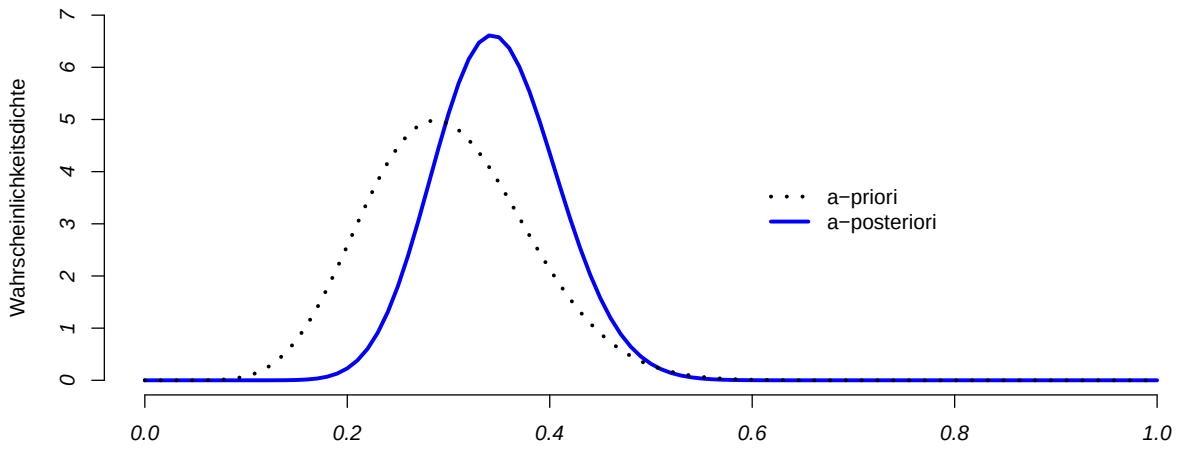
```
tab <- cbind(p, prior, round(Likel, 4), round(posterior, 4))
colnames(tab) <- c("p","a-priori","Likelihood","a-posteriori")
as.data.frame(tab)
```

p	a-priori	Likelihood	a-posteriori
0.05	0.05	0.0000	0.0000
0.15	0.15	0.0006	0.0016
0.25	0.30	0.0291	0.1592
0.35	0.30	0.1254	0.6868
0.45	0.05	0.1265	0.1155
0.55	0.05	0.0379	0.0346
0.65	0.04	0.0031	0.0022
0.75	0.02	0.0000	0.0000
0.85	0.02	0.0000	0.0000
0.95	0.02	0.0000	0.0000

6.19.1 A-priori Verteilungen (Prior)

Beispiel Langschläfer:

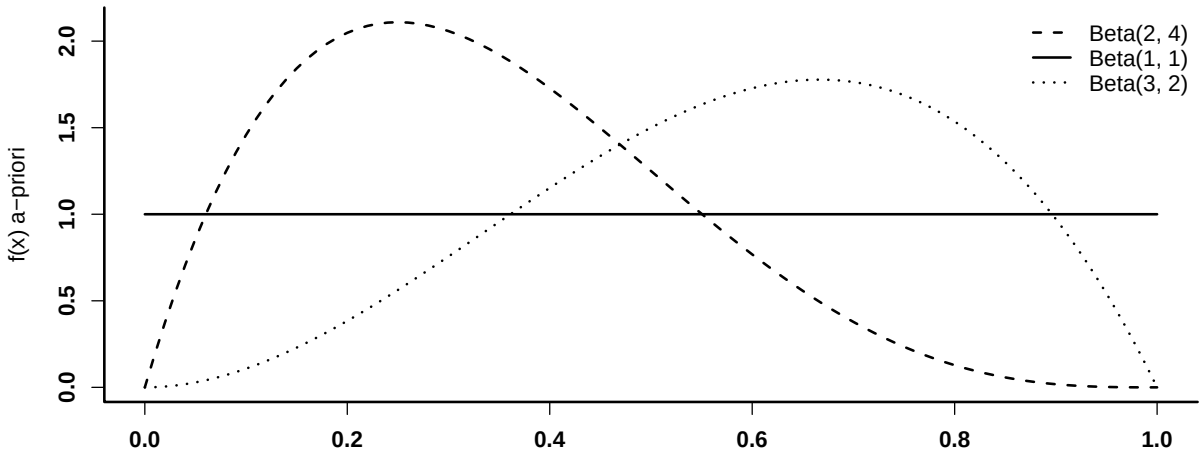
```
obs <- c(0.20, 0.25, 0.30, 0.35, 0.40)                          # a-priori Wissen (?)

m <- mean(obs); m;  s <- sd(obs); s                             # Beta-Verteilung (prior)
## [1] 0.3
## [1] 0.07905694
p.0 <- m * (m*(1-m)/s^2 - 1); p.0
## [1] 9.78
q.0 <- (1-m) * (m*(1-m)/s^2 -1); q.0
## [1] 22.82

prob    <- seq(0, 1, by=0.01)
prior     <- dbeta(prob, p.0, q.0)                              # a-prior Dichte

n <- 30; x <- 12                                                # Beobachtung (Stichprobe)

p.1 <- p.0 + x; p.1; q.1 <- q.0 +n - x; q.1
## [1] 21.78
## [1] 40.82
posterior <- dbeta(prob, p.1, q.1)                              # a-posterior Dichte

par(mfrow=c(1,1), lwd=2, bty="n", font=1, font.axis=3, font.lab=1, ps=12)
plot(prob, posterior, typ="l", ylab="Wahrscheinlichkeitsdichte",
              ylim=c(0,7), xlab=" ", lty=1, lwd=3, col=4)
lines(prob, prior, lty=3, lwd=3, col=1)
legend(.6, 4, c("a-priori","a-posteriori"), lty=c(3,1),
               lwd=c(3,3), col=c(1,4), bty="n")
```



6.19.2 Parameterschätzung nach Bayes

Beispiel faire oder gefälschte Münze:

```
par(mfcol=c(1,1), lwd=2, font.axis=2, bty="l", ps=13)
x <- seq(0, 1, 0.01)
f1 <- dbeta(x, 2, 4, ncp = 0, log = FALSE)
f2 <- dbeta(x, 1, 1, ncp = 0, log = FALSE)
f3 <- dbeta(x, 3, 2, ncp = 0, log = FALSE)
plot(x, f1, type = "l", lty=2, xlim=c(0,1), xlab=" ", ylab="f(x) a-priori")
lines(x, f2, type = "l", lty=1, xlim=c(0,1))
lines(x, f3, type = "l", lty=3, xlim=c(0,1))
legend("topright", legend = c("Beta(2, 4)","Beta(1, 1)","Beta(3, 2)"),
               lty = c(2,1,3), xjust = 1, yjust = 1, bty="n")
```




```
                                                    # Parameter-Schaetzung
alpha  <- 0.10; target <- 1 - alpha
qbeta(alpha/2, p.1, q.1)                            # CR - credible interval
## [1] 0.2524403
qbeta(1-alpha/2, p.1, q.1)
## [1] 0.4489757
```

```
region <- c(0.1, 0.6)                               # Laufzeit ...?
ruler  <- seq(region[1], region[2], length=10000)
dens   <- round(dbeta(ruler, p.1, q.1), 2)          # a-posteriori Dichte

start  <- 1; stop <- length(dens)                   # HPD - Region
done   <- FALSE; tol <- 0.01
i <- start;
while (i < stop & done == FALSE) {
    j <- length(dens)
    while (j > i & done == FALSE) {
```

```
        if (dens[i] == dens[j]) {
            L <- pbeta(ruler[i], p.1, q.1)
            H <- pbeta(ruler[j], p.1, q.1)
            if (((H - L) < (target+tol)) & ((H - L) > (target-tol)))
                done <- TRUE
            }
            j <- j - 1
        }
        i <- i + 1
}
k <- dens[i]; HPD.L <- ruler[i]; HPD.H <- ruler[j]
k; HPD.L; HPD.H
## [1] 1.63
## [1] 0.2466647
## [1] 0.4486849

par(mfrow=c(1,1), lwd=2, bty="n", font=1, font.axis=3, font.lab=1, ps=12)
plot(prob, posterior, typ="l", ylab="Wahrscheinlichkeitsdichte",
                        ylim=c(0,7), xlab=" ", lty=1, lwd=3, col=4)
abline(h=k)
lines(c(HPD.L, HPD.L), c(0, dens[i]))
lines(c(HPD.H, HPD.H), c(0, dens[j]))
text(0.85, k + 0.3, paste("k = ",k))
text(0.1, 1, paste("HPD (L):", round(HPD.L, 3)), ps=10)
text(0.6, 1, paste("HPD (H):", round(HPD.H, 3)), ps=10)
```