AS17: Kapitel 3 - Deskriptive Statistik

Jürgen Hedderich

2020-04-13

Contents

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:

Zuverlässigkeit: reliabilität

Elektrischer Widerstand: resistivity

Nichtparametr. Regression: nonparegdat

3.1 Häufigkeiten

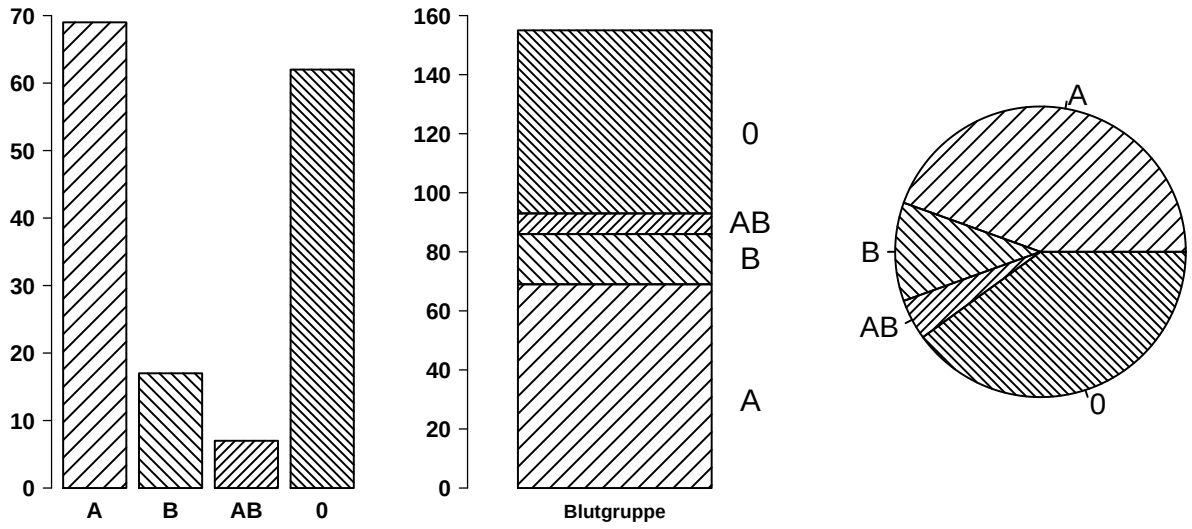
3.1.1 Absolute und relative Häufigkeiten

```r
                                                  # Blutgruppen A, B, AB, 0
Blutgruppen <- c(rep("A", 69), rep("B", 17), rep("AB", 7), rep("0", 62))
vt <- table(Blutgruppen); vt
## Blutgruppen
##  0  A AB  B
## 62 69  7 17
Modalwert <- (names(vt[vt == max(vt)])); Modalwert
## [1] "A"

vect <- sample(0:9, size=25, replace=T); vect
##  [1] 5 0 9 7 8 8 7 3 4 9 6 4 7 9 9 4 6 3 1 1 9 6 9 2 8
vt <- table(vect); mode <- as.numeric(names(vt[vt == max(vt)])); mode
## [1] 9

absolut  <- c(69, 17, 7, 62)                      # Blutgruppen A, B, AB, 0
names(absolut) <- c("A","B","AB","0"); absolut
##  A  B AB  0
## 69 17  7 62
anzahl   <- sum(absolut); anzahl
## [1] 155
relativ  <- absolut / anzahl; round(relativ, 2)
##    A    B   AB    0
## 0.45 0.11 0.05 0.40
prozent  <- relativ * 100; round(prozent, 1)
##    A    B   AB    0
## 44.5 11.0  4.5 40.0

Gini     <- sum(relativ*(1-relativ)); Gini        # Gini-Simpson-Index
## [1] 0.6277627
```

3.1.4 Balken und Kreisdiagramme

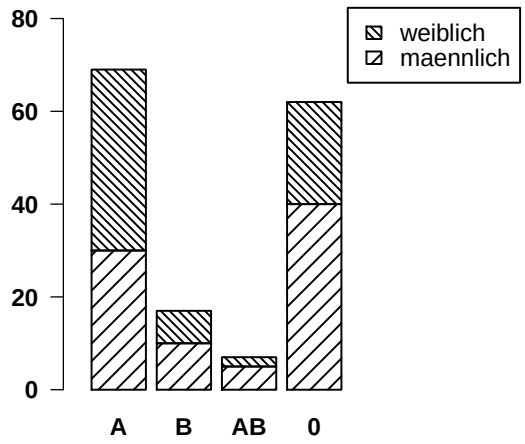
Funktion barplot2() in library(gplots)

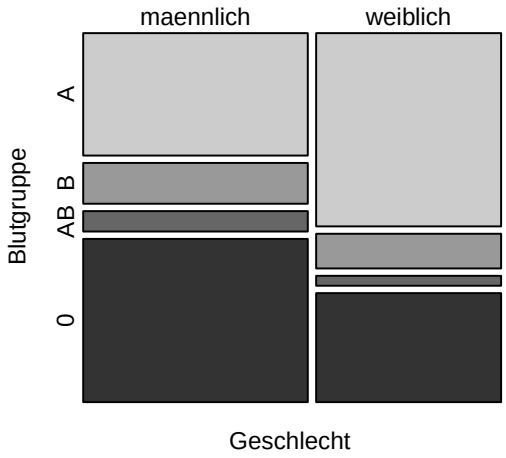
```
absolut  <- c(69, 17, 7, 62)                          # Blutgruppen A, B, AB, 0
names(absolut) <- c("A","B","AB","0"); absolut
##  A  B AB  0
## 69 17  7 62


                                                      # Torten- und Balkendiagramm
library(gplots)
par(mfrow=c(1,3), lwd=1.5, font.axis=2, bty="n", ps=15, cex.axis=1)
barplot2(absolut, names.arg=c("A","B","AB","0"), las=1,
        cex.axis = 1.3, cex.names = 1.3, ylim=c(0,70),
        density=c(10,15,18,20), angle=c(45,135,45,135), col="black")
barplot2(as.matrix(absolut), names.arg="Blutgruppe", beside = FALSE,
        ylim=c(0,160), yaxp=c(0,160,8), xlim=c(0,1.5), las=1,
        cex.axis = 1.3, cex.names = 1.1,
        density=c(10,15,18,20),angle=c(45,135,45,135), col="black")
text(1.4,30,"A",bg="white",cex=1.8); text(1.4,78,"B",bg="white",cex=1.8)
text(1.4,90,"AB",bg="white",cex=1.8); text(1.4,120,"0",bg="white",cex=1.8)
pie(absolut, labels=c("A","B","AB","0"), radius = 1.0,
    density=c(10,15,18,20),angle=c(45,135,45,135), col="black", cex=1.7)
```



3.1.5 Rechteckdiagramm und Mosaikplot

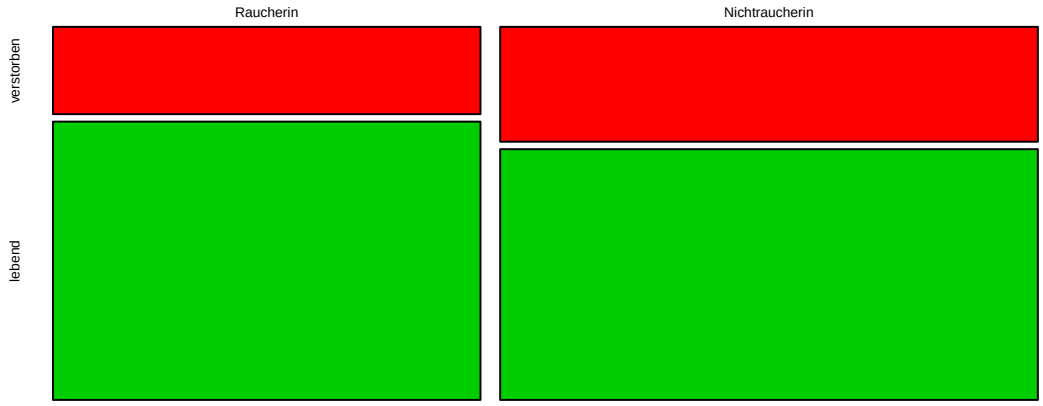
```
                                                    # Blutgruppen nach dem Geschlecht
absolut <- matrix(c(30, 10, 5, 40, 39, 7, 2, 22), nrow=2, byrow=T)
colnames(absolut)        <- c("A","B", "AB","0")
rownames(absolut)        <- c("maennlich", "weiblich")
names(dimnames(absolut)) <- c("Geschlecht","Blutgruppe"); absolut
##            Blutgruppe
## Geschlecht   A  B AB  0
##   maennlich 30 10  5 40
##   weiblich  39  7  2 22
margin.table(absolut, 1)
## Geschlecht
## maennlich  weiblich
##        85        70
margin.table(absolut, 2)
## Blutgruppe
##  A  B AB  0
## 69 17  7 62
round(prop.table(absolut, 1), 3)
##            Blutgruppe
## Geschlecht      A     B    AB     0
##   maennlich 0.353 0.118 0.059 0.471
##   weiblich  0.557 0.100 0.029 0.314
round(prop.table(absolut, 2), 3)
##            Blutgruppe
## Geschlecht      A     B    AB     0
##   maennlich 0.435 0.588 0.714 0.645
##   weiblich  0.565 0.412 0.286 0.355

par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="n", ps=14, cex.axis=1)
barplot(absolut, density=c(10, 20), col="black", angle=c(45,135),
        las=1, legend.text=T, ylim=c(0,85), xlim=c(0,8))
mosaicplot(absolut,  col=c("grey80","grey60","grey40","grey20"),
           cex=1.0, main=" ")
```


80
60
40
20
0
A
B
AB
0
weiblich
maennlich


maennlich
weiblich
Blutgruppe
A
B
AB
0
Geschlecht

3.1.6 Bedingte Häufigkeiten (Simpson's Paradox)

```
                                                    # Simpson-Paradox (Anteilswerte)
data <- array(c(2,53,1,61, 3,121,5,152, 14,95,7,114, 27,103,12,66,
                51,64,40,81, 29,7,101,28, 13,0,64,0), dim=c(2,2,7),
        dimnames=list(c("verstorben","lebend"),
                      c("Raucherin","Nichtraucherin"),
        c("18-24","25-35","35-44","45-54","55-64","65-74",">74")))
tsums <- apply(data,c(1,2),sum)

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="n", ps=12, cex.axis=2)
mosaicplot(t(tsums), main=" ", col=2:3)
```




```
rbind(tsums, round(tsums[1,]/apply(tsums,2,sum)*100, 1))
##            Raucherin Nichtraucherin
## verstorben     139.0          230.0
## lebend         443.0          502.0
##                 23.9           31.4
for (i in 1:7) { t <- data[,,i]; apply(t,2,sum)
print(cbind(t,round(t[1,]/apply(t,2,sum)*100,1))) }
##            Raucherin Nichtraucherin
## verstorben         2              1 3.6
## lebend            53             61 1.6
##            Raucherin Nichtraucherin
## verstorben         3              5 2.4
## lebend           121            152 3.2
##            Raucherin Nichtraucherin
## verstorben        14              7 12.8
## lebend            95            114  5.8
##            Raucherin Nichtraucherin
## verstorben        27             12 20.8
```

```
## lebend           103             66 15.4
##            Raucherin Nichtraucherin
## verstorben        51             40 44.3
## lebend            64             81 33.1
##            Raucherin Nichtraucherin
## verstorben        29            101 80.6
## lebend             7             28 78.3
##            Raucherin Nichtraucherin
## verstorben        13             64 100
## lebend             0              0 100
```

```
                                                    # Assoziationsmaße; Goodman-Kruskal
tau_GK <- function(dat) {
    N <- sum(dat);
    dat.rows <- nrow(dat);  dat.cols <- ncol(dat)
    max.col <- sum.col <- L.col <- matrix(,dat.cols)
    max.row <- sum.row <- L.row <- matrix(,dat.rows)
    for(i in 1:dat.cols) sum.col[i] <- sum(dat[,i]); max.col[i] <- max(dat[,i])
    for(i in 1:dat.rows) sum.row[i] <- sum(dat[i,]); max.row[i] <- max(dat[i,])
    max.row.margin <- max(apply(dat,1,sum));
    max.col.margin <- max(apply(dat,2,sum));
    p.a <- N^2                      # tau Column|Row
    for(i in 1:dat.rows) p.a <- p.a - N * sum(dat[i,]^2/sum.row[i])
    p.b <- N^2 - sum(sum.col^2);
    tau.CR <- 1 - (p.a / p.b)
    p.a <- N^2                      # tau Row|Column
    for(j in 1:dat.cols) p.a <- p.a - N * sum(dat[,j]^2/sum.col[j])
    p.b <- N^2-sum(sum.row^2)
    tau.RC <- 1 - (p.a / p.b)
    cat("\ntau_Col:Row = ",round(tau.CR, 3),
    " and tau_Row:Col = ",round(tau.RC, 3),"\n")
}

x <- matrix(c(10,30,5,0,20,30,5,0,0), byrow=T, nrow=3); x
##      [,1] [,2] [,3]
## [1,]   10   30    5
## [2,]    0   20   30
## [3,]    5    0    0
tau_GK(x)
##
## tau_Col:Row =  0.236  and tau_Row:Col =  0.28
```

3.2 Ordinaldaten

3.2.1 Medianwert und Quantile

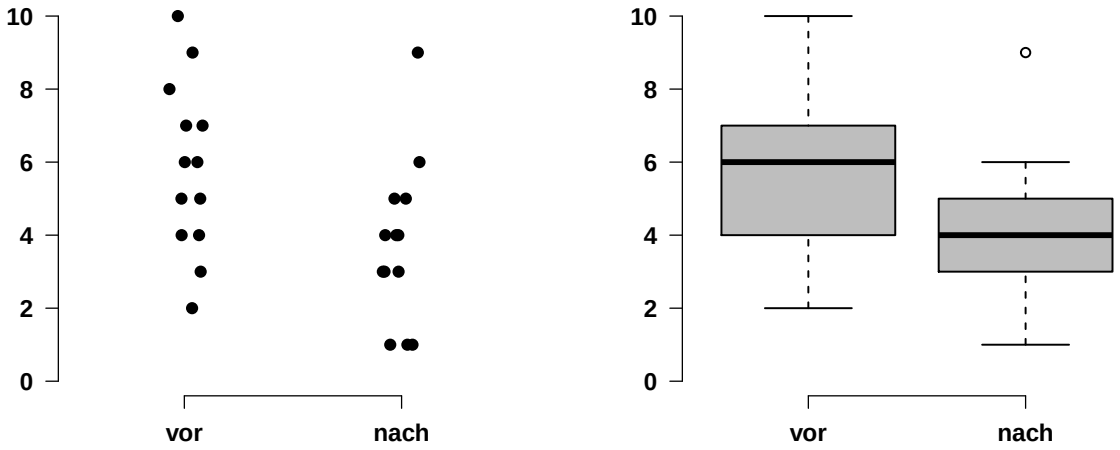
```
                                                        # Rangdaten
vor  <- c(3,  4,  6,  4,  8,  9,  2,  7, 10,  7,  5,  6,  5 )
nach <- c(4,  4,  1,  5,  3,  3,  1,  3,  4,  5,  6,  9,  1 )

vor; sort(vor)
##  [1]  3  4  6  4  8  9  2  7 10  7  5  6  5
##  [1]  2  3  4  4  5  5  6  6  7  7  8  9 10
vor; rank(vor)
##  [1]  3  4  6  4  8  9  2  7 10  7  5  6  5
##  [1]  2.0  3.5  7.5  3.5 11.0 12.0  1.0  9.5 13.0  9.5  5.5  7.5  5.5

                                                        # Quartile
vor  <- c(3,  4,  6,  4,  8,  9,  2,  7, 10,  7,  5,  6,  5 )
vsort <- sort(vor); n  <- length(vsort)
Q1       <- vsort[floor((n+1)*0.25)]; Q1
## [1] 4
Q2       <- vsort[floor((n+1)*0.50)]; Q2
## [1] 6
Q3       <- vsort[floor((n+1)*0.75)]; Q3
## [1] 7

median(vor);
## [1] 6
quantile(vor, c(0.25, 0.50, 0.75))
## 25% 50% 75%
##   4   6   7
```

3.2.3 Streuung ordinal skalierter Daten

```
MA <- mean(abs(vor-median(vor))); MA                    # Median-Absolutabweichung
## [1] 1.846154

D  <- mad(vor, const=1); D                              # Median-Deviation
## [1] 2
```

3.2.4 Punktdiagramm und Boxplot

```
                                                        # Dot-Plot  und Box-Plot
vor  <- c(3,  4,  6,  4,  8,  9,  2,  7, 10,  7,  5,  6,  5 )
nach <- c(4,  4,  1,  5,  3,  3,  1,  3,  4,  5,  6,  9,  1 )

par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="n", ps=14, cex.axis=1)
stripchart(list(vor, nach), method="jitter", jitter=0.1, las=1,
           vertical=TRUE, group.names=c("vor","nach"),
           xlim=c(0.5,2.5), ylim=c(0,10), pch=16, cex=1.3)
boxplot(vor, nach, range = 1.5, names=c("vor","nach"),
        las=1, ylim=c(0,10), col=8)
```



3.2.5 Korrelationskoeffizient nach Kendall

```
kendall.tau <- function(x, y) {                          # Kendall Korrelationskoeffizient
  n <- length(x)
  if(n != length(y)) stop("x und y unterschiedliche L?nge")
  x <- rank(x);    y <- rank(y)                          # Rangzahlen !
  m <- cbind(x,y); m <- m[order(m[,1]),]                # Ordnung nach x
  x <- m[,1];      y <- m[,2]
  inv <- 0; prov <- 0
  for (i in 1:n) {
    for (j in i:n) {
      if (x[i]<x[j] & y[i]>y[j]) inv  <- inv + 1        # Inversion
      if (x[i]<x[j] & y[i]<y[j]) prov <- prov + 1       # Proversion
    }  }
  rx <- table(x); Tx <- 0.5*sum(rx*(rx-1))              # Bindungen in x
  ry <- table(y); Ty <- 0.5*sum(ry*(ry-1))              # Bindungen in y
  r.tau_b = (prov-inv)/sqrt((prov + inv + Tx)*(prov + inv + Ty))
  cat("Anzahl Proversionen:",prov,"und Inversionen:",inv,"\n",
      "Kendalls Tau_b:",r.tau_b,"\n")
}

x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)                   # Weinsorten
y <- c(2, 1, 5, 3, 4, 6, 7, 9, 8, 10)                   # ohne Bindungen
kendall.tau(x , y)
## Anzahl Proversionen: 41 und Inversionen: 4
##  Kendalls Tau_b: 0.8222222


x <- c( 8,  5, 10, 10,  8); x; rank(x)                  # Rangzahlen, ordnen
## [1]  8  5 10 10  8
## [1] 2.5 1.0 4.5 4.5 2.5
y <- c(15, 12,  6, 16, 12); y; rank(y)                  # mit Bindungen
## [1] 15 12  6 16 12
## [1] 4.0 2.5 1.0 5.0 2.5
kendall.tau(x , y)
## Anzahl Proversionen: 4 und Inversionen: 3
##  Kendalls Tau_b: 0.1178511
cor(x, y , method = "kendall")                          # spez. Funktion cor()
## [1] 0.1178511
```

3.2.6 Partielle Rangkorrelation

```
z <- c(1,2,3,4,5,6)                                    # einführendes Beispiel
x <- c(3,1,4,2,6,5)
y <- c(4,2,1,6,3,5)

tauXZ <- round(cor(x, z, method="kendall"), 4)
tauYZ <- round(cor(y, z, method="kendall"), 4)
tauXY <- round(cor(x, y, method="kendall"), 4)
tauXY.Z <- (tauXY - tauXZ*tauYZ) / sqrt((1-tauXZ^2)*(1-tauYZ^2))
cbind(tauXZ, tauYZ, tauXY, tauXY.Z)
##       tauXZ tauYZ   tauXY    tauXY.Z
## [1,] 0.4667   0.2 -0.0667 -0.1846871

                                                       # partielle Rangkorrelation
I <- c(1,2,3,4,5,6,7,8,9,10)                           # Intelligenz
A <- c(1,4,5,6,2,7,3,9,8,10)                           # Mathematik
B <- c(4,1,3,5,2,6,7,10,9,8)                           # Musik

tauAI <- round(cor(A, I, method="kendall"), 4)
tauBI <- round(cor(B, I, method="kendall"), 4)
tauAB <- round(cor(A, B, method="kendall"), 4)
tauAB.I <- (tauAB - tauAI*tauBI) / sqrt((1-tauAI^2)*(1-tauBI^2))
cbind(tauAI, tauBI, tauAB, tauAB.I)
##       tauAI  tauBI  tauAB   tauAB.I
## [1,] 0.6444 0.6444 0.5556 0.2400153
```

3.3 Beschreibung metrischer Daten

3.3.1 Arithmetischer Mittelwert

```
                                                    # Body-Mass-Index
bmi <- c(28.2,23.9,20.3,26.7,25.6,32.5,23.5,19.7,27.8,26.7,20.7,28.4,33.3)
n     <- length(bmi)
Summe <- sum(bmi); Summe
## [1] 337.3
Summe/n                                             # arithmetisches Mittel
## [1] 25.94615

mean(bmi)                                           # Funktion mean()
## [1] 25.94615

bmi <- c(22.2,23.9,20.3,26.7,25.6,22.5,23.5,24.7,27.8,26.7,20.7,26.4,40.3)
sort(bmi)
##  [1] 20.3 20.7 22.2 22.5 23.5 23.9 24.7 25.6 26.4 26.7 26.7 27.8 40.3
mean(bmi)
## [1] 25.48462
mean(bmi, trim=0.1)                                 # gestutzter Mittelwert
## [1] 24.60909

winsor <- function(x, tr=.1) {                      # winsorisierter Mittelwert
    if(tr < 0 || tr > 0.5) stop("Anteil?")
    y <- sort(x)
    n <- length(x)
    ibot <- floor(tr*n)+1; itop <- length(x)-ibot+1
    xbot <- y[ibot];       xtop <- y[itop]
    y <- ifelse(y<=xbot, xbot, y)
    y <- ifelse(y>=xtop, xtop, y)
    mean(y)
}

bmi <- c(22.2,23.9,20.3,26.7,25.6,22.5,23.5,24.7,27.8,26.7,20.7,26.4,40.3)
mean(bmi)
## [1] 25.48462
winsor(bmi)
## [1] 24.55385
```

3.3.2 Standardabweichung, Varianz

```
bmi <- c(28.2,23.9,20.3,26.7,25.6,32.5,23.5,19.7,27.8,26.7,20.7,28.4,33.3)
m     <- mean(bmi)
saq   <- (bmi - m)^2
sqrt(sum(saq)/(n-1))                                    # Standardabweichung
## [1] 4.295466

sd(bmi)                                                 # Funktion sd()
## [1] 4.295466
var(bmi)                                                # Funktion var()
## [1] 18.45103
```

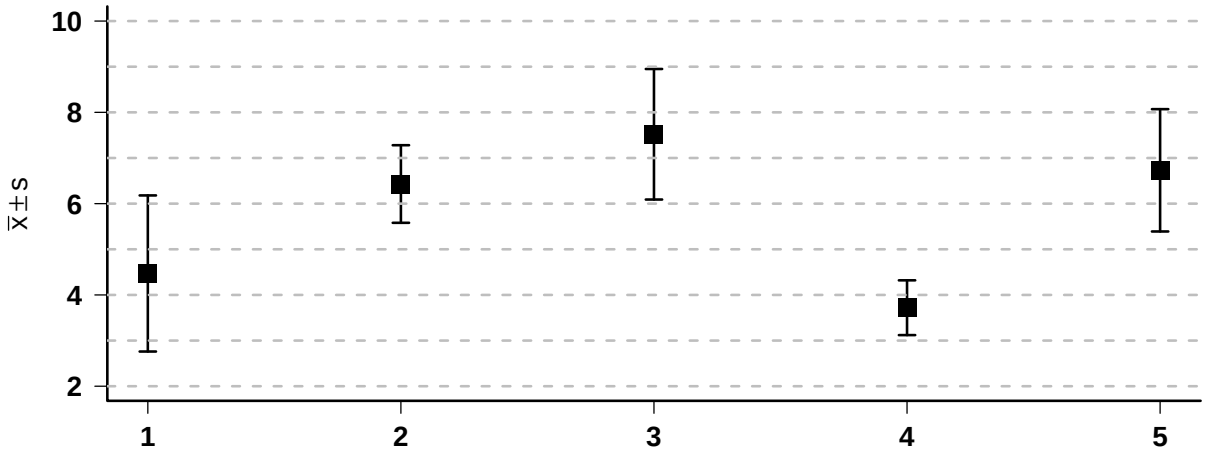
3.3.3 Kombination von Mittelwerten und Varianzen

```
s1 <- c(40,50,72); s2 <- c(30,60,80,90,100);  s3 <- c(40,50,60,70)
n <- c(3, 5, 4)
m <- c(mean(s1), mean(s2), mean(s3))
s <- c(sd(s1), sd(s2), sd(s3)); v <- s^2

mv.combi <- function(n, m, v) {                         # Funktion nach Yiu-Man Chan
    r   <- length(n)
    n.t <- sum(n);                  m.t <- sum(n*m) / n.t
    ts1 <- sum((n-1)*v);    ts2 <- 0
    for (k in 1:(r-1)) for (l in (k+1):r)
        ts2 <- ts2 + n[k]*n[l]*(m[k]-m[l])^2
    v.t <- (ts1 + (1/n.t)*ts2) / (n.t-1)
    cat("\n","Summe =",n.t," - Mittelwert =",m.t," - Varianz =",v.t)
}

mv.combi(n=c(3, 5, 4), m=c(54, 72, 55), v=c(267.98, 770.06, 166.67))
##
##  Summe = 12  - Mittelwert = 61.83333  - Varianz = 454.8979
```

3.3.5 Streubereich um den Mittelwert

Funktion errbar() in library(Hmisc)

```
library(Hmisc)

x     <- 1:5
y     <- c(4.47, 6.43, 7.52, 3.72, 6.73)
delta <- c(1.71, 0.85,  1.43, 0.6,  1.34)

par(mfrow=c(1,1), lwd=2.0, font.axis=2, bty="l", ps=16, cex.axis=1)
errbar( x, y, y + delta, y - delta , ylim=c(2, 10), xlab=" ", pch=15,
        las=1, lwd=2.0, cex=2.0, ylab=expression(bar(x)%+-%s))
abline(h=seq(2,10,1), lty=2, col="grey")
```



3.3.7 Gewogener / gewichteter Mittelwert

```
gew.stats <- function(meanis, ni, varis=NULL) {
  k <- length(meanis);     n <- sum(ni)
  mean <- sum(ni*meanis)/n
  if (is.null(varis)) {
    mgew <- sum(ni*meanis)/n
    cat("\n","Gewogener Mittelwert:", round(mgew, 2),"\n")
  }
  if (!is.null(varis)) {
    mgew <- sum(ni*meanis/varis)/sum(ni/varis)
    sin  <- sqrt(sum(varis*(ni-1)) / (n-k))
    vgew <- (sum((ni-1)*varis) + sum(ni*(meanis - mean)^2)) / (n-1)
    cat("\n","Gewogener Mittelwert  :", round(mgew, 2),"\n",
        "Standardabw. innerhalb:", round(sin, 3),"\n",
        "Gewogene Varianz      :", round(vgew, 3),"\n")
  }
}

n <- c(8, 10, 6); m <- c(9,  7, 8); s <- c(2,  1, 2)    # gewogener Mittelwert bei ...
gew.stats(m, n)                                          # gleiche Varianzen
##
##  Gewogener Mittelwert: 7.92
gew.stats(m, n, s^2)                                     # ungleiche Varianzen
##
##  Gewogener Mittelwert  : 7.41
##  Standardabw. innerhalb: 1.648
##  Gewogene Varianz      : 3.254


note <- c(  2,   3,   3,   2,   1)
gew  <- c(0.3, 0.3, 0.2, 0.1, 0.1)
weighted.mean(note, gew)                                 # gewichteter Mittelwert
## [1] 2.4
```

3.3.8 Geometrischer Mittelwert / Wachstum

```
                                                    # Seventy rule (70iger Regel)
rt <- 1:10                                          # Wachstumsraten (%)
n_rule  <- round(70/rt, 2)
n_exact <- round(log(2)/log(1+(rt/100)), 2)
tab <- cbind(rt, n_rule, n_exact); tab
##       rt n_rule n_exact
##  [1,]  1  70.00   69.66
##  [2,]  2  35.00   35.00
##  [3,]  3  23.33   23.45
##  [4,]  4  17.50   17.67
##  [5,]  5  14.00   14.21
##  [6,]  6  11.67   11.90
##  [7,]  7  10.00   10.24
##  [8,]  8   8.75    9.01
##  [9,]  9   7.78    8.04
## [10,] 10   7.00    7.27
                                                    # geometrischer Mittelwert
gehalt    <- c(1.025, 1.10, 1.22)                   # Gehaltserhöhungen
lg.gehalt <- log10(gehalt)
10^mean(lg.gehalt)                                  # mittlere Gehaltserhoehung
## [1] 1.112138
```

3.3.9 Harmonischer Mittelwert / Kosten

```
stueck     <- c(10, 5, 8)                                  # Kosten / Stueckzahl
rez.stueck <- 1/stueck; n <- length(stueck)
n / sum(rez.stueck)                                        # mittlere Stueckzahl
## [1] 7.058824
```

3.4 Fehlerrechnung

3.4.2 Standardfehler bei Mehrfachbestimmung

```
mat <- matrix(c(11, 13, 12, 27, 25, 29, 43, 47, 42, 63, 57, 60), nrow=4, byrow=T)
mat                                                        # Beispiel Dreifachbestimmung
##      [,1] [,2] [,3]
## [1,]   11   13   12
## [2,]   27   25   29
## [3,]   43   47   42
## [4,]   63   57   60
Bmean <- apply(mat, 1, mean); Bmean
## [1] 12 27 44 60
Bsaq  <- apply(mat, 1, function(x) sum((x - mean(x))^2)); Bsaq
## [1]  2  8 14 18
sMB <- sqrt(sum(Bsaq) / 8); sMB
## [1] 2.291288
Pmean <- apply(mat, 2, mean); Pmean
## [1] 36.00 35.50 35.75
Psaq <- apply(mat, 2, function(x) sum((x - mean(x))^2)); Psaq
## [1] 1484.00 1211.00 1236.75
Pv   <- Psaq/3; Pv
## [1] 494.6667 403.6667 412.2500
quotient <- sMB / sqrt(sum(Pv)/3); quotient
## [1] 0.1096246
```

3.4.4 Präzision von Messungen

```
                                                    # elektrischer Widerstand
resist <- read.csv("resistivity.csv", header = TRUE, sep = ";", dec=",", fill = FALSE)
attach(resist); resist[1:10, ]
```

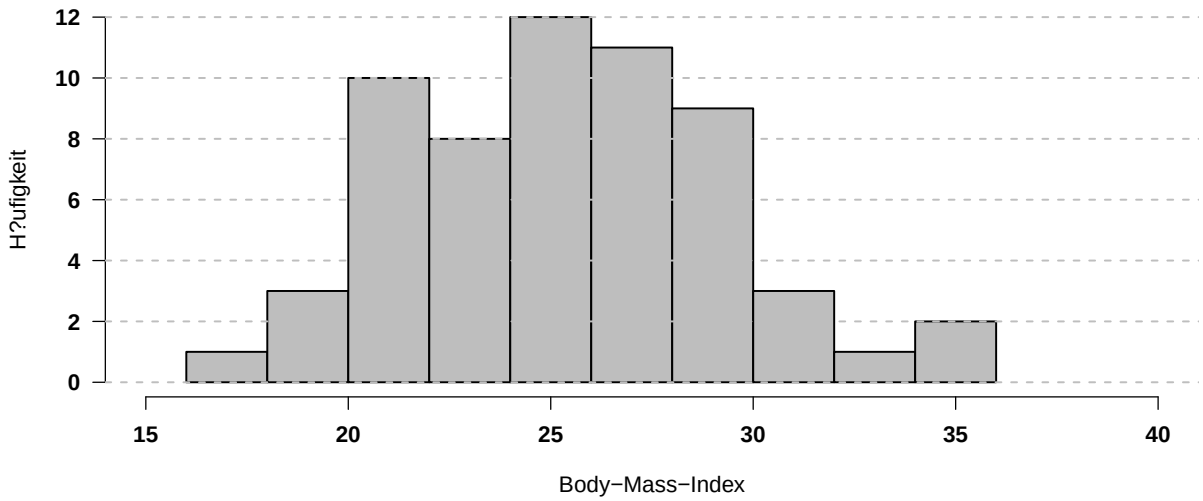
nr	Serie	Tag	Anzahl	Mittelwert	Stdabw
1	1	1	6	96.0771	0.1024
2	1	2	6	95.9976	0.0943
3	1	3	6	96.0148	0.0622
4	1	4	6	96.0397	0.0702
5	1	5	6	96.0407	0.0627
6	1	6	6	96.0445	0.0622
7	2	1	6	96.0793	0.0996
8	2	2	6	96.1115	0.0533
9	2	3	6	96.0803	0.0364
10	2	4	6	96.0411	0.0768

```
N <- 6                                              # Wiederholungen
L <- 2                                              # Serien
K <- 6                                              # Tage

                                        # Wiederholbarbeit in der Serie gepoolt
s1 <- sqrt(sum(resist$Stdabw^2)/(K*L)); round(s1, 4)
## [1] 0.0787

                                        # Reproduzierbarbeit von Tag zu Tag und gepoolt
s2.1 <- sd(resist$Mittelwert[Serie==1]); round(s2.1, 4)
## [1] 0.0273

s2.2 <- sd(resist$Mittelwert[Serie==2]); round(s2.2, 4)
## [1] 0.0276

s2 <- sqrt((s2.1^2 + s2.2^2)/L); round(s2, 4)
## [1] 0.0274
                                        # Stabilität / Konstanz von Serie zu Serie
m1 <- mean(resist$Mittelwert[Serie==1])
m2 <- mean(resist$Mittelwert[Serie==2])
s3 <- sd(c(m1, m2)); round(s3, 4)
## [1] 0.0288
                                        # Fehler / Unsicherheit für ein einzelnes Ergebnis
s.R <- sqrt(s3^2 + ((K-1)/K)*s2^2 + ((N-1)/N)*s1^2); round(s.R, 4)
## [1] 0.0814
```

3.5 Häufigkeitsverteilung

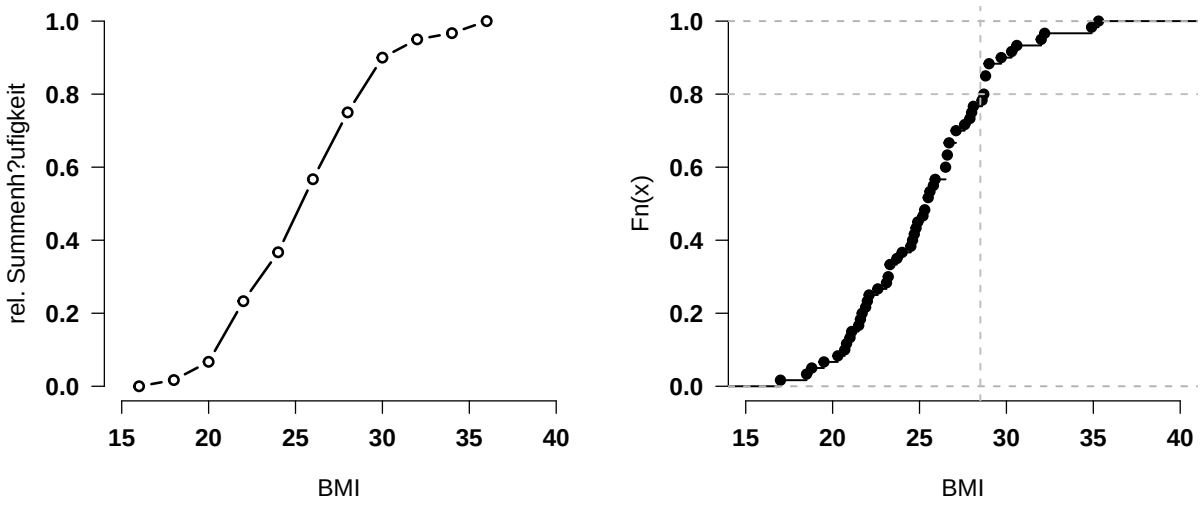
3.5.1 Histogramm / Summenhäufigkeitsverteilung

```
bmi <- c(20.8, 29.7, 27.6, 28.6, 20.7, 21.0, 23.1, 21.9, 24.8, 25.3, 27.1, 23.3,
         19.5, 25.2, 25.8, 21.6, 28.7, 30.6, 23.3, 26.6, 35.3, 17.0, 22.6, 25.9,
         29.0, 23.7, 21.7, 26.5, 18.5, 24.5, 29.0, 23.2, 27.9, 18.8, 27.1, 21.5,
         26.5, 20.3, 25.5, 32.0, 26.7, 34.9, 24.6, 25.6, 26.7, 22.1, 28.8, 28.1,
         28.8, 32.2, 30.3, 24.9, 28.0, 21.1, 22.0, 25.5, 24.0, 26.6, 24.7, 28.8)

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=11, cex.axis=1)
hist(bmi, breaks=c(16, 18,20,22,24,26,28,30,32,34,36), col="grey", las=1,
     cex.axis=1.2, cex.lab=1.2,
     xlim=c(15,40), xlab="Body-Mass-Index", ylab="H?ufigkeit", main=" ")
abline(h=seq(0,12,2,), lty=2, col="grey")
```

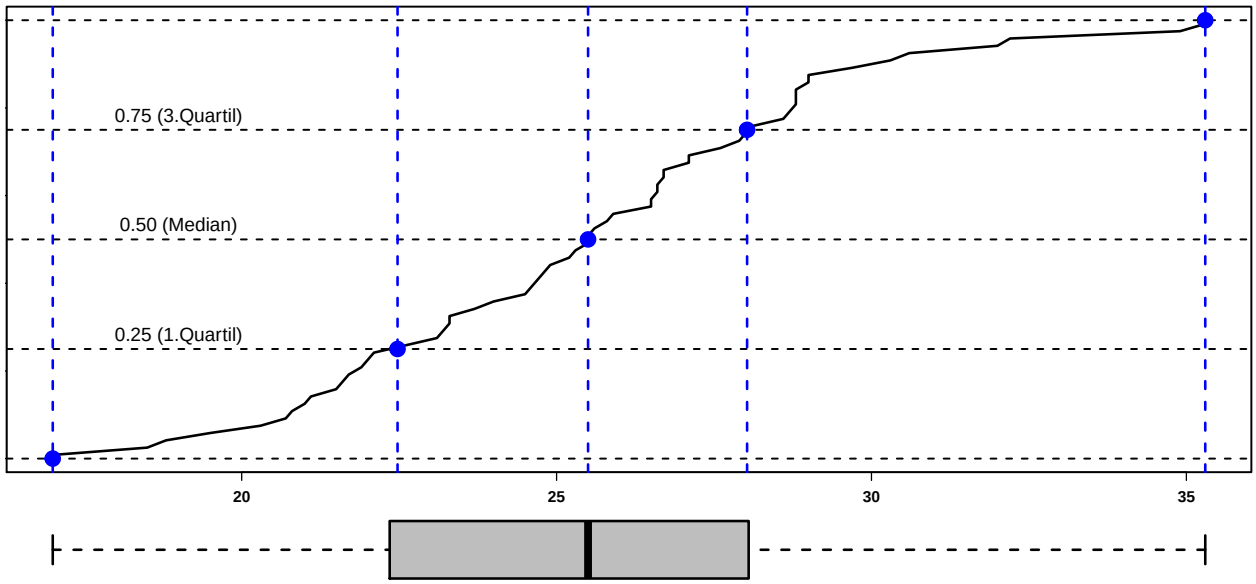



```
h <- hist(bmi, breaks=c(16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36), plot=FALSE)
h$breaks; h$mids; h$counts; cumsum(h$counts)
##  [1] 16 18 20 22 24 26 28 30 32 34 36
##  [1] 17 19 21 23 25 27 29 31 33 35
##  [1]  1  3 10  8 12 11  9  3  1  2
##  [1]  1  4 14 22 34 45 54 57 58 60

round(h$counts/length(bmi),3)*100
##  [1]  1.7  5.0 16.7 13.3 20.0 18.3 15.0  5.0  1.7  3.3

round(cumsum(h$counts)/length(bmi),3)*100
##  [1]   1.7   6.7  23.3  36.7  56.7  75.0  90.0  95.0  96.7 100.0

h <- hist(bmi, breaks=c(16, 18,20,22,24,26,28,30,32,34,36), plot=FALSE)
par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="n", ps=14, cex.axis=1)
```

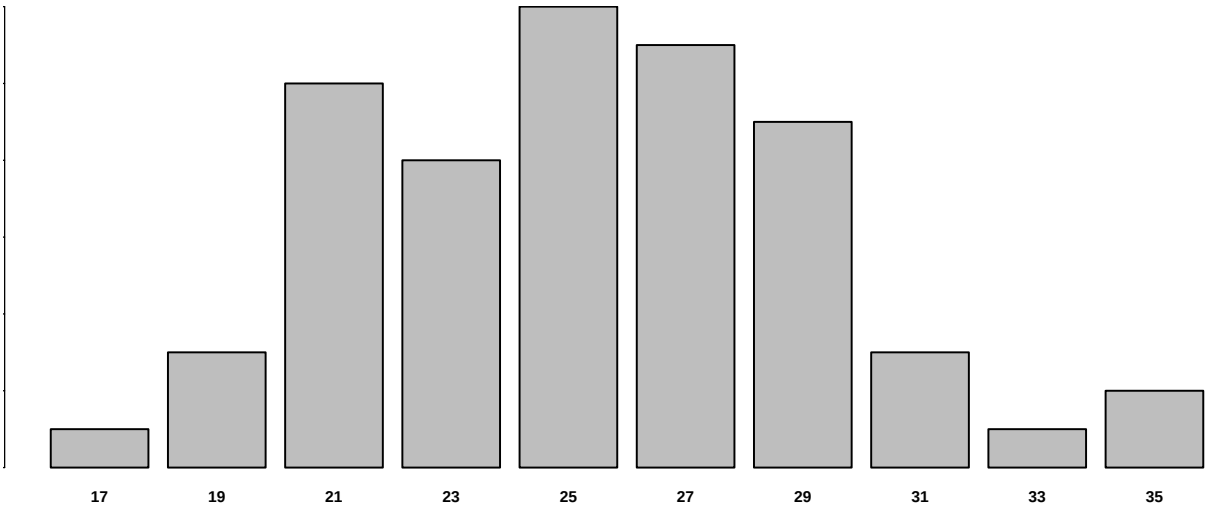
```
x <- h$breaks;  y <- c(0, round(cumsum(h$counts)/length(bmi),3))
plot(x, y, type="b", ylim=c(0,1), xlim=c(15,40), las=1, lwd=2,
          xlab="BMI", ylab="rel. Summenh?ufigkeit" )
plot.ecdf(bmi, xlab="BMI", col.vert = "gray20", las=1, xlim=c(15,40), main=" ")
abline(h=0.8, lty=2, col="grey")
abline(v=28.5, lty=2, col="grey")
```




```
par(lwd=1.5, font.axis=2, ps=14, cex.axis=1, bty="o")
op <- par(mar=c(0,0,0,0), oma=c(0,0,0,0)+.1)
layout(matrix(c(1,1,1,2), nc=1))
plot(sort(bmi), ppoints(length(bmi)), type="l", lwd=2,
           xlab="", ylab="", main="", xaxt="n")
axis(1, labels = TRUE, tick = TRUE)
text(19,0.28,"0.25 (1.Quartil)", cex=1.2)
text(19,0.53,"0.50 (Median)", cex=1.2)
text(19,0.78,"0.75 (3.Quartil)", cex=1.2)
abline(h = c(0,.25,.5,.75,1), lty=2)
abline(v = quantile(bmi), col = "blue", lwd = 2, lty=2)
points(quantile(bmi), c(0,.25,.5,.75,1), lwd= 8, col="blue")
boxplot(bmi, horizontal = TRUE, col = "grey",
           lwd=2, cex=1.5, pch=16, axes = FALSE)
```

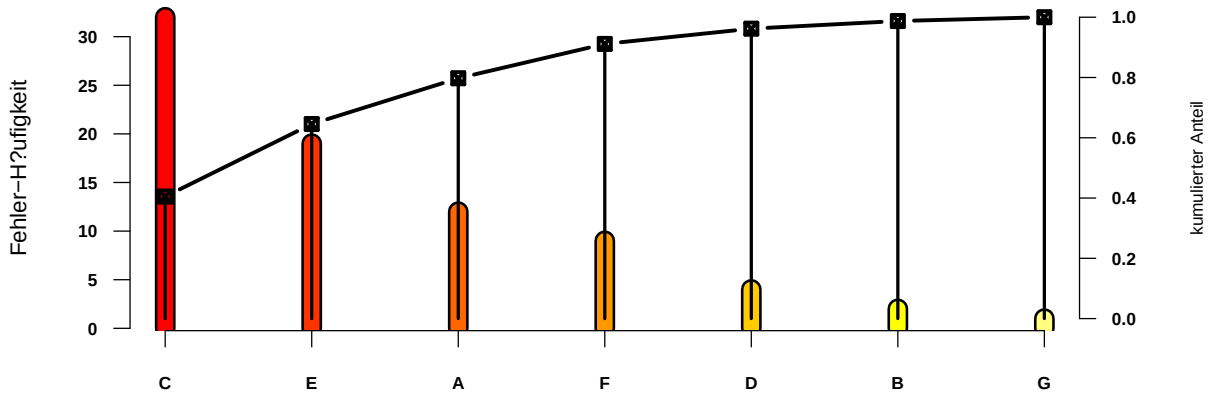



```
quantil.hist <- function(p, breaks, freq) {                # Quantile aus klassierten Daten
  n <- sum(freq); rel <- freq/n
  if(any(p<=0) | any(p>=1)) return("Fehler: 0<p<1")
  cum <- cumsum(c(0, rel))
  loc <- apply(outer(p, cum, ">="), 1, sum)
  return(breaks[loc] + (p - cum[loc])/rel[loc] *
           (breaks[loc+1] - breaks[loc]))   }

ecdf.hist <- function(x, breaks, freq) {                   # Anteile aus klassierten Daten
  n <- sum(freq); rel <- freq/n
  if(x < breaks[1]) return("Fehler: x < min")
  else if(x > breaks[length(breaks)])
    return("Fehler: x > max")
  else {loc <- max(which(breaks <= x))
  return(sum(rel[1:(loc-1)])+(x-breaks[loc])/
           (breaks[loc+1] - breaks[loc]) * rel[loc])}   }


breaks <- c(16, 18, 20, 22, 24, 26, 28, 30, 32, 34)
mids   <- c(17, 19, 21, 23, 25, 27, 29, 31, 33, 35)
counts <- c( 1,  3, 10,  8, 12, 11,  9,  3,  1,  2)

barplot(counts, names.arg=mids, las=1, xlab="Body-Mass-Index")
quantil.hist(c(0.25, 0.50, 0.75), breaks, counts)        # approximiert
## [1] 22.25000 25.33333 28.00000
quantile(bmi, probs=c(0.25, 0.50, 0.75))                 # exakt
##    25%    50%    75%
## 22.475 25.500 28.025

ecdf.hist(x=30, breaks, counts)
## [1] 0.9
```


17
19
21
23
25
27
29
31
33
35

3.5.2 Pareto-Diagramm

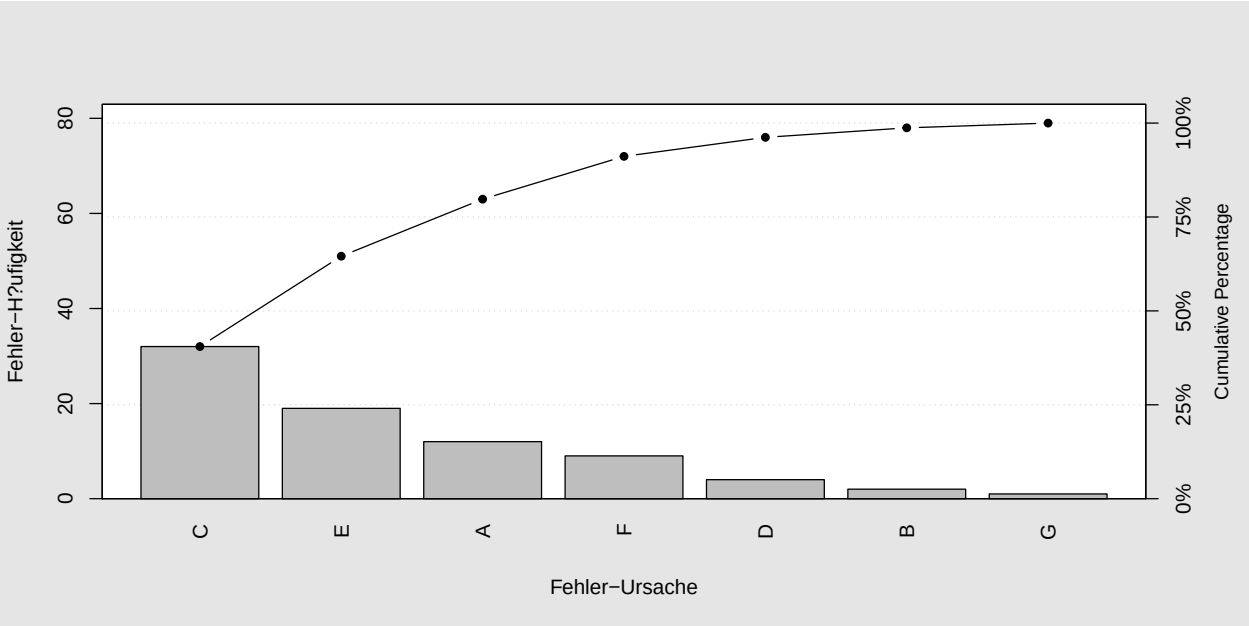
```
pareto <- function (x, main = "", ylab = "Value") {
  op <- par(mar = c(5, 4, 4, 5) + 0.1, las = 2)
  x <- rev(sort(x))
  plot( x, type = 'h', axes = F, lwd = 16,
                  xlab = "", ylab = ylab, main = main )
  axis(2)
  points( x, type = 'h', lwd = 12, col = heat.colors(length(x)) )
  y <- cumsum(x)/sum(x)
  par(new = T)
  plot(y, type = "b", lwd = 3, pch = 7, axes = FALSE,
       xlab='', ylab='', ylim=c(0,1), main='')
  points(y, type = 'h')
  axis(4)
  par(las=0)
  mtext("kumulierter Anteil", side=4, line=3)
  print(names(x))
  axis(1, at=1:length(x), labels=names(x))
  par(op)
}

defect <- c(12, 2, 32, 4, 19, 9, 1)
names(defect) <- c("A", "B", "C", "D", "E", "F", "G")

par(mfrow=c(1,1), lwd=2.5, font.axis=2, bty="u", ps=10, cex.axis=0.8, cex=1.3)
pareto(defect, ylab="Fehler-H?ufigkeit")
```




```
## [1] "C" "E" "A" "F" "D" "B" "G"
```

Funktion pareto.chart() in library(qcc)

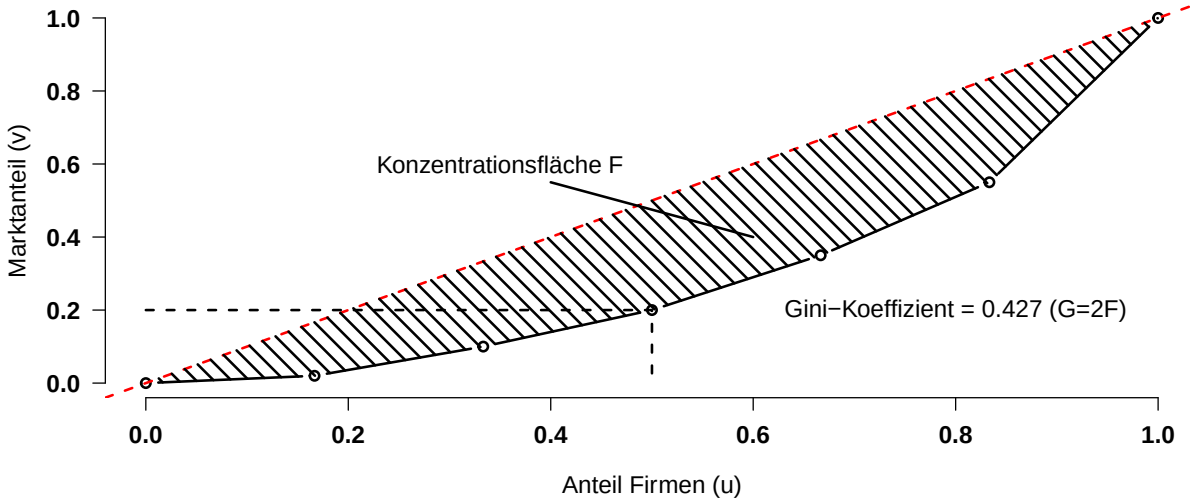
```
library(qcc)                                          # pareto.chart() in library(qcc)
pareto.chart(defect, ylab = "Fehler-H?ufigkeit", main =" ",
                  xlab="Fehler-Ursache", las=3, col="grey")
```




```
##
## Pareto chart analysis for defect
##      Frequency  Cum.Freq. Percentage Cum.Percent.
##   C  32.000000  32.000000  40.506329    40.506329
##   E  19.000000  51.000000  24.050633    64.556962
##   A  12.000000  63.000000  15.189873    79.746835
##   F   9.000000  72.000000  11.392405    91.139241
##   D   4.000000  76.000000   5.063291    96.202532
##   B   2.000000  78.000000   2.531646    98.734177
##   G   1.000000  79.000000   1.265823   100.000000
```

3.6 Konzentration - Herfindahl-Index und Gini-Koeffizient

```
gini <- function(x, y) {                                    # Funktion zum Gini-Koeffizient
    area <- 0                                               # Trapezregel
    for (i in 2:(n+1)) area <- area + 0.5*((x[i]-x[i-1])*(y[i]+y[i-1]))
    gini <- 1 - 2*area; round(gini, 3)                      # Gini-Koeffizient
}
x <- c(2, 8, 10, 15, 20, 45);  n <- length(x)
u <- c(0, (1:n)/n)                                          # Abszisse - relativer Index
v <- c(0, (cumsum(x) / sum(x)))                             # Odinate  - kuml. rel. Anteile
gini(u, v)
## [1] 0.427


                                                            # Lorenzkurve und Gini-Koeffizient
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="n", ps=14)
x <- c(2, 8, 10, 15, 20, 45);  n <- length(x)
u <- c(0, (1:n)/n)                                          # Abszisse - relativer Index
v <- c(0, (cumsum(x) / sum(x)))                             # Odinate  - kumul. rel. Anteile
plot(u, v, type="b", cex=1.0, xlim=c(0,1), ylim=c(0,1), las=1,
     xlab="Anteil Firmen (u)",  ylab="Marktanteil (v)")
abline(0,1, col="red", lty=2)
text(0.8, 0.20, paste("Gini-Koeffizient =",round(gini(u, v),3),"(G=2F)"))
text(0.35, 0.6, "Konzentrationsfläche F")
lines(c(0.4,0.6),c(0.55,0.4), lwd=2)
lines(c(0,0.5,0.5), c(0.2,0.2,0), lty=2)
polygon(c(u,0), c(v,0), angle = -45, border=NA, density = 10)
```

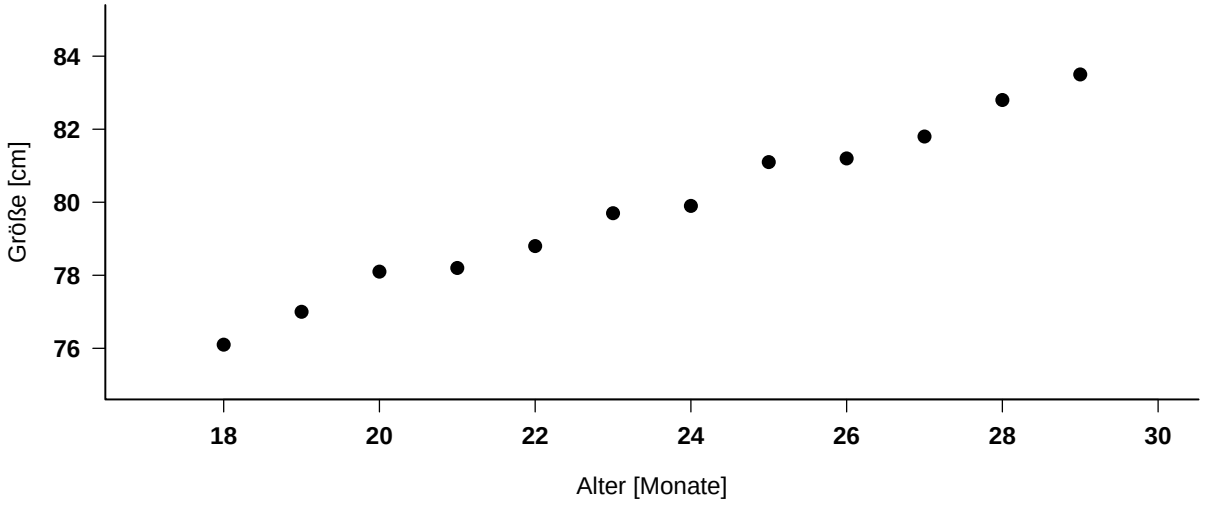



```
gini_num <- function(x, unbiased =FALSE) {              # Gini-Koeffizient
N    <- length(x)
mu   <- mean(x)
n    <- if (unbiased) N * (N-1) else N * N
ox   <- x[order(x)]
dsum <- drop(crossprod(2 * 1:N - N - 1, ox))
round(dsum / (mu * n), 3)
}

x <- c(2, 8, 10, 15, 20, 45); gini_num(x, unbiased=FALSE)
## [1] 0.427
```

3.7 Maßzahlen für den Zusammenhang metrischer Daten

3.7.2 Punktwolken

```
x <- seq(18, 29, by=1)                                    # Alter und Körpergröße in Kalama
y <- c(76.1, 77.0, 78.1, 78.2, 78.8, 79.7, 79.9, 81.1, 81.2, 81.8, 82.8, 83.5)

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=14, cex.axis=1)
plot(x, y, pch=16, cex=1.5, las=1,
     xlab="Alter [Monate]", ylab="Größe [cm]",
     xlim=c(17, 30), ylim=c(75, 85))
```

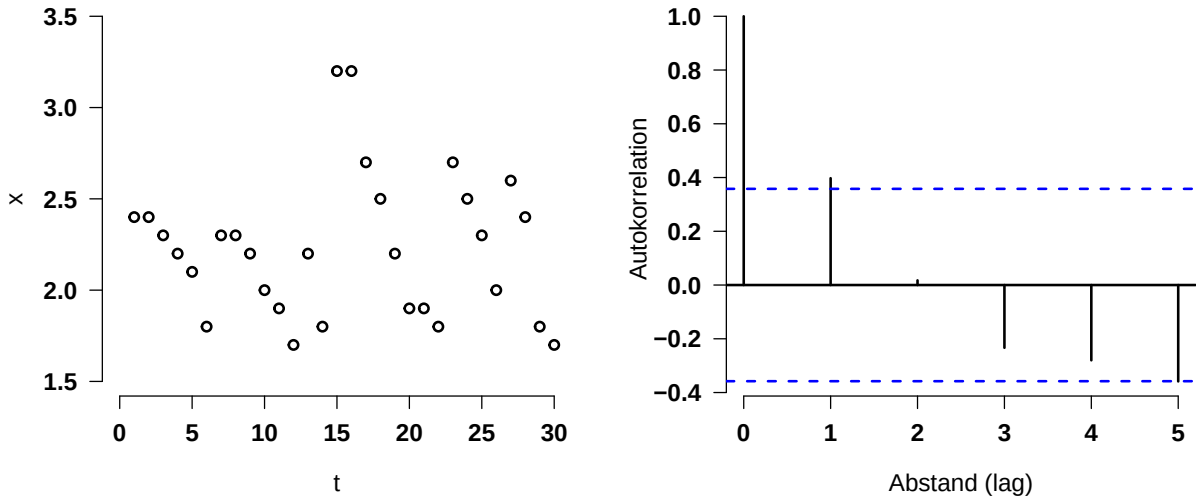


3.7.3 Empische Kovarianz / Empirischer Korrelationskoeffizient

```
x <- c(13, 17, 10, 17, 20, 11, 15)
y <- c(12, 17, 11, 13, 16, 14, 15)

var(x); var(y);                                           # Varianz
## [1] 12.90476
## [1] 4.666667
cov(x, y)                                                 # Kovarianz
## [1] 5.5
cor(x, y)                                                 # Korrelationskoeffizient
## [1] 0.7087357
```

3.7.5 Autokorrelation

```
x <- c(2.4, 2.4, 2.3, 2.2, 2.1, 1.8, 2.3, 2.3, 2.2, 2.0, 1.9, 1.7, 2.2, 1.8, 3.2,
       3.2, 2.7, 2.5, 2.2, 1.9, 1.9, 1.8, 2.7, 2.5, 2.3, 2.0, 2.6, 2.4, 1.8, 1.7)
n <- length(x)
t <- seq(1, n, by=1)
acorr <- acf(x, lag.max=5, plot=FALSE); acorr
##
## Autocorrelations of series 'x', by lag
##
##      0      1      2      3      4      5
##  1.000  0.398  0.017 -0.234 -0.280 -0.359

par(mfrow=c(1,2), lwd=2, font.axis=2, bty="n", ps=14)
plot(t, x, ylim=c(1.5,3.5), xlim=c(0,n), las=1)
acorr <- acf(x, lag.max=5, plot=FALSE)
plot(acorr, main=" ", las=1, ylab="Autokorrelation", xlab="Abstand (lag)")
```



3.7.6 Reliabilitätsanalyse

```
# fiktive Beispieldaten
test <- read.csv("reliabilität.csv", header = TRUE, sep = ";")
attach(test); test[1:5, ]
```

proband	test	item1	item2	item3	item4	item5	item6
1	1	3	4	2	3	4	2
2	1	2	3	1	2	3	2
3	1	2	2	1	1	2	1
4	1	3	3	3	2	3	2
5	1	4	4	2	3	3	4

```
k <- 6                                                  # Anzahl der Items

cor(score, r_score)                                     # Test-Retest Reliabilitaet
## [1] 0.8404992

                                                        # Split-Half
test1 <- cbind(item1, item2, item3); score1 <- apply(test1, 1, sum)
test2 <- cbind(item4, item5, item6); score2 <- apply(test2, 1, sum)

r_tt  <- cor(score1, score2); r_tt                      # Reliabilitaet
## [1] 0.763246

r_tt1 <- 2*r_tt / (1+r_tt); r_tt1                       # Spearman-Brown Korrektur
## [1] 0.8657283

                                                        # Guttmann Koeffizient
r_tt2 <- 2 * (1 - (var(score1) + var(score2))/var(score)); r_tt2
## [1] 0.8539877

mat <- cbind(item1, item2, item3, item4, item5, item6)
si  <- sum(diag(cov(mat)))                              # Item Varianzen
st  <- var(score)                                       # Gesamtvarianz im Score

alpha <- k/(k-1) * (1 - si/st); alpha                   # Cronbach's Alpha
## [1] 0.7038037
```

Funktion alpha() in library(psych)

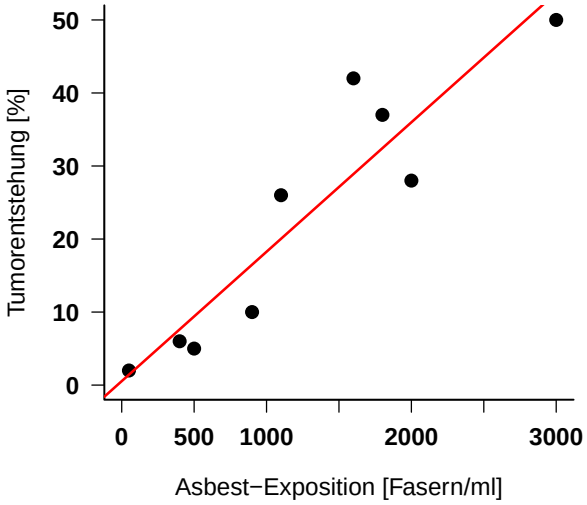
```
library(psych)
alpha(mat)                                              # Funktion alpha() in library(psych)
##
## Reliability analysis
## Call: alpha(x = mat)
##
##   raw_alpha std.alpha G6(smc) average_r S/N  ase mean   sd median_r
##        0.7      0.73    0.91      0.32 2.8 0.17  2.4 0.64     0.41
##
##  lower alpha upper     95% confidence boundaries
## 0.37 0.7 1.04
##
##  Reliability if an item is dropped:
##       raw_alpha std.alpha G6(smc) average_r S/N alpha se var.r med.r
## item1      0.55      0.61    0.88      0.24 1.5     0.27 0.134  0.32
## item2      0.63      0.68    0.86      0.30 2.1     0.21 0.118  0.39
## item3      0.70      0.74    0.89      0.36 2.8     0.18 0.133  0.49
## item4      0.55      0.56    0.83      0.20 1.3     0.26 0.126  0.18
## item5      0.71      0.74    0.89      0.37 2.9     0.17 0.083  0.45
## item6      0.78      0.79    0.90      0.43 3.7     0.13 0.083  0.49
##
##  Item statistics
##       n raw.r std.r r.cor r.drop mean   sd
## item1 8  0.85  0.86  0.82  0.738  2.6 1.06
## item2 8  0.71  0.71  0.69  0.524  2.6 1.06
## item3 8  0.53  0.54  0.48  0.316  2.0 0.93
## item4 8  0.94  0.94  0.95  0.910  2.0 0.76
## item5 8  0.51  0.52  0.48  0.274  2.9 0.99
## item6 8  0.39  0.37  0.32  0.096  2.2 1.16
##
## Non missing response frequency for each item
##          1    2    3    4 miss
## item1 0.12 0.38 0.25 0.25    0
## item2 0.12 0.38 0.25 0.25    0
## item3 0.38 0.25 0.38 0.00    0
## item4 0.25 0.50 0.25 0.00    0
## item5 0.12 0.12 0.50 0.25    0
## item6 0.25 0.50 0.00 0.25    0
```

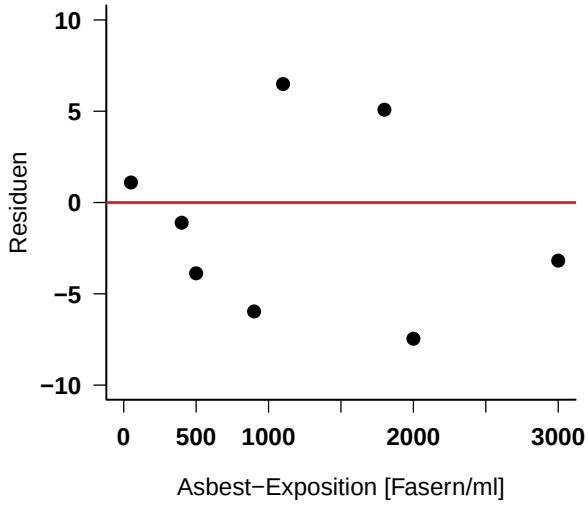
3.7.7 Rangkorrelation nach Spearman

```
L <- c(1, 2, 2, 2, 3, 3, 4, 4); M <- c(2, 4, 1, 3, 4, 3, 4, 3)
r.L <- rank(L, ties.method="average"); r.L            # Rangzahlen zu x
## [1] 1.0 3.0 3.0 3.0 5.5 5.5 7.5 7.5
r.M <- rank(M, ties.method="average"); r.M            # Rangzahlen zu y
## [1] 2 7 1 4 7 4 7 4

D <- r.L - r.M; n <- length(D)
1- 6*sum(D^2)/(n*(n^2-1))                          # Rangkorrelationskoeffizient (Spearman)
## [1] 0.5357143
cor(r.L, r.M)                                      # Korrelationskoeffizient aus Rangzahlen
## [1] 0.4935481
```

3.7.9 Lineare Regression

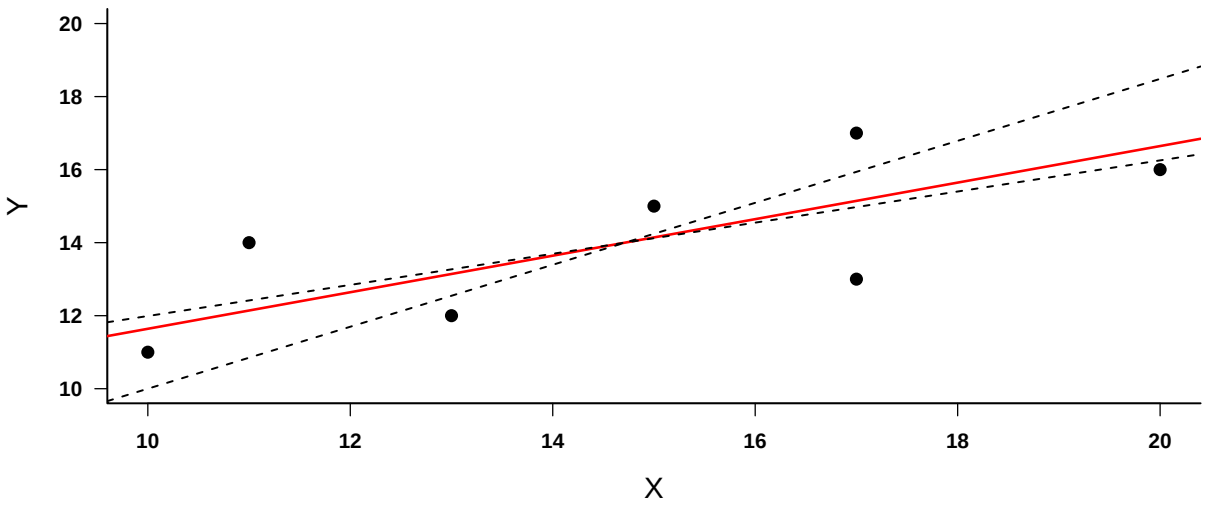
```
                                                   # Asbest und Auftreten von Lungentumoren
asbest <- c(50, 400, 500, 900, 1100, 1600, 1800, 2000, 3000)
lungca <- c(2, 6, 5, 10, 26, 42, 37, 28, 50)

lm(lungca ~ asbest)
##
## Call:
## lm(formula = lungca ~ asbest)
##
## Coefficients:
## (Intercept)       asbest
##     0.54047      0.01772
                                                   # lineare Regression
reg <- lm(lungca ~ asbest); reg$coefficients
## (Intercept)      asbest
##  0.54046893  0.01772121
lungca.hat <- reg$coefficients[2] + reg$coefficients[2]*asbest
resid      <- lungca - lungca.hat

par(mfrow=c(1,2), lwd=1.5, font.axis=2, bty="l", ps=14, cex.axis=1)
plot(asbest, lungca, pch=16, cex=1.5, las=1, ylab="Tumorentstehung [%]",
     xlab="Asbest-Exposition [Fasern/ml]", ylim=c(0, 50), xlim=c(0,3000))
abline(reg$coefficients[1], reg$coefficients[2], col="red", lwd=2)
plot(asbest, resid, cex=1.5, las=1, xlim=c(0, 3000), pch=16, ylim=c(-10, ++10),
     xlab="Asbest-Exposition [Fasern/ml]", ylab="Residuen")
abline(h=0, col="red", lwd=2)
```


Tumorentstehung [%]
50
40
30
20
10
0
0
500
1000
2000
3000
Asbest−Exposition [Fasern/ml]


Residuen
10
5
0
−5
−10
0
500
1000
2000
3000
Asbest−Exposition [Fasern/ml]

3.7.10 Orthogonale Regression

```r
x <- c(13, 17, 10, 17, 20, 11, 15)
y <- c(12, 17, 11, 13, 16, 14, 15)
                                                    # orthogonale Regression
Q.x  <- sum((x - mean(x))^2); Q.y  <- sum((y - mean(y))^2)
Q.xy <- sum((x - mean(x))*(y - mean(y)))

b <- (-(Q.x-Q.y)+sqrt((Q.x-Q.y)^2+4*Q.xy^2)) / (2 * Q.xy); b
## [1] 0.5004332
a <- mean(y) - b*mean(x); a
## [1] 6.636483

r1 <- lm(y ~ x); a1 <- r1$coeff[1]; b1 <- r1$coeff[2]
r2 <- lm(x ~ y); a2 <- r2$coeff[1]; b2 <- r2$coeff[2]

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l",
    cex.lab=1.4, ps=12, cex.axis=1)
plot(x, y, pch=16, cex=1.4, ylab="Y", xlab="X", las=1,
      ylim=c(10, 20), xlim=c(10, 20))
abline(a, b, col="red", lwd=2)
abline(a1, b1, lty=2)
abline(-a2/b2, 1/b2, lty=2)
```



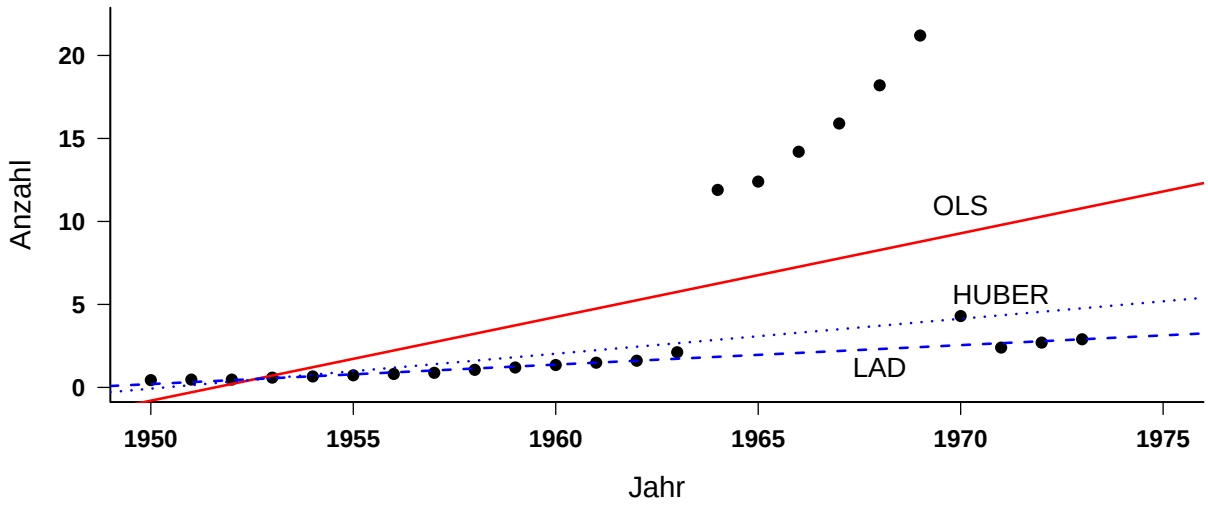
3.7.11 Robuste lineare Regression

Funktion rq() in library(quantreg)

Funktion rlm() in library(MASS)

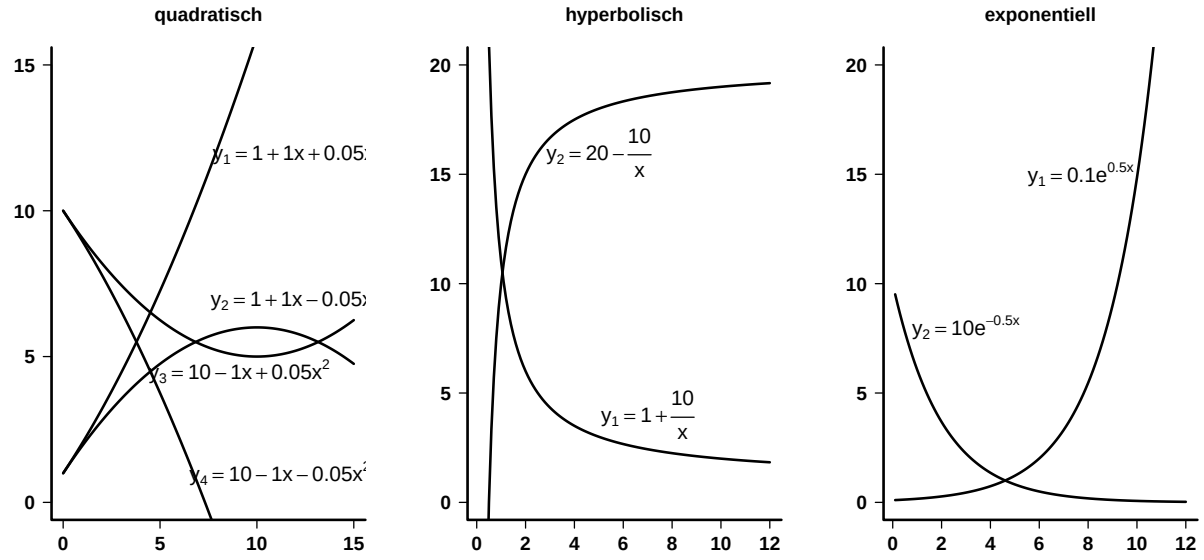
```
library(quantreg)
library(MASS)

Jahr <- seq(1950, 1973, by=1)           #  internationale Telephonate in Belgien 1950 - 1973
Anzahl <- c(0.44,0.47,0.47,0.59,0.66,0.73,0.81,0.88,1.06,1.2,1.35,1.49,1.61,
            2.12,11.90,12.40,14.20,15.90,18.20,21.20,4.30,2.40,2.70,2.90)

OLS.regr <- lm(Anzahl ~ Jahr); OLS.regr
##
## Call:
## lm(formula = Anzahl ~ Jahr)
##
## Coefficients:
## (Intercept)         Jahr
##   -983.8868       0.5041

LAD.regr <- rq(Anzahl ~ Jahr, tau=0.5); LAD.regr
## Call:
## rq(formula = Anzahl ~ Jahr, tau = 0.5)
##
## Coefficients:
##  (Intercept)         Jahr
## -228.4790909    0.1172727
##
## Degrees of freedom: 24 total; 22 residual

HUBER.regr <- rlm(Anzahl ~ Jahr, psi = psi.huber, k2 = 1.345); HUBER.regr
## Call:
## rlm(formula = Anzahl ~ Jahr, psi = psi.huber, k2 = 1.345)
## Ran 20 iterations without convergence
##
## Coefficients:
##  (Intercept)         Jahr
## -410.5620825    0.2105068
##
## Degrees of freedom: 24 total; 22 residual
## Scale estimate: 0.985

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=12,
    cex.lab=1.4, cex.axis=1.2)
plot(Jahr, Anzahl, pch=16, cex=1.3, ylab="Anzahl", las=1,
       xlab="Jahr", ylim=c(0, 22), xlim=c(1950,1975))
abline(OLS.regr$coefficients[1],
       OLS.regr$coefficients[2],   col="red", lwd=2, lty=1)
text(1970, 11, "OLS", cex=1.3)
abline(LAD.regr$coefficients[1],
       LAD.regr$coefficients[2],   col="blue", lwd=2, lty=2)
text(1968, 1.2, "LAD", cex=1.3)
```

```
abline(HUBER.regr$coefficients[1],
       HUBER.regr$coefficients[2], col="blue", lwd=2, lty=3)
text(1971, 5.6, "HUBER", cex=1.3)
```

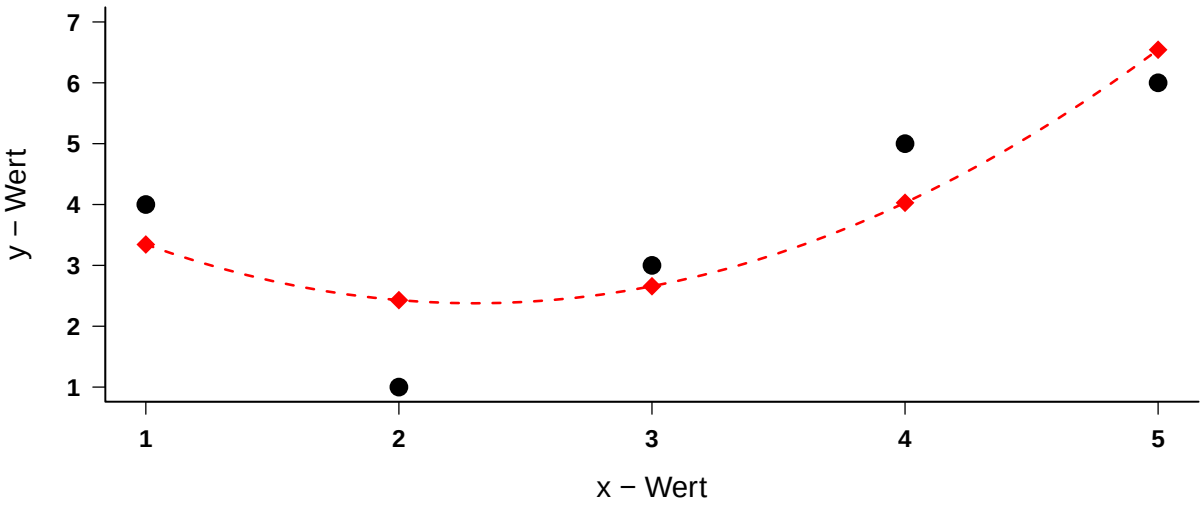


3.7.12 Nichtlineare Regression

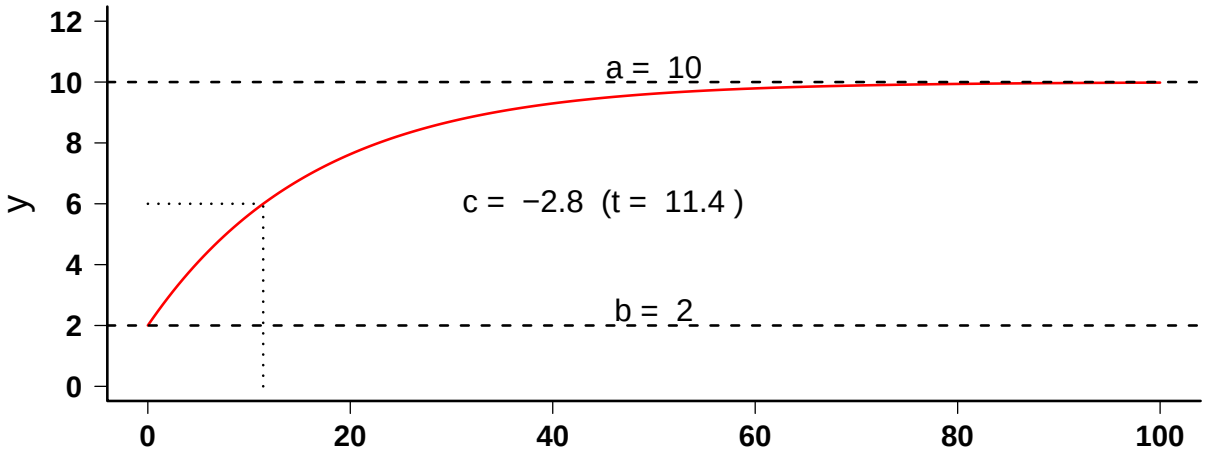
```
par(mfrow=c(1,3), lwd=2, font.axis=2, bty="l", ps=14,
    cex.axis=1.2, cex.lab=1.4)
x <- seq(0, 15, by=0.1)
y1 <-  1 + 1*x + 0.05*x^2; y2 <-  1 + 1*x - 0.05*x^2
y3 <- 10 - 1*x + 0.05*x^2; y4 <- 10 - 1*x - 0.05*x^2
plot(x, y1, type ="l", main="quadratisch", xlab=" ", las=1,
      ylab=" ", ylim=c(0,15))
lines(x, y2); lines(x, y3); lines(x, y4)
text(12.1, 12, expression(y[1] == 1 + 1*x + 0.05*x^2), cex=1.3)
text(12,  7, expression(y[2] == 1 + 1*x - 0.05*x^2), cex=1.3)
text( 9.1, 4.5, expression(y[3] == 10 - 1*x + 0.05*x^2), cex=1.3)
text(11.2,  1, expression(y[4] == 10 - 1*x - 0.05*x^2), cex=1.3)

x <- seq(0.1, 12, by=0.1); y1 <- 1  + 10/x; y2 <- 20 - 10/x
plot(x, y1, type="l", main="hyperbolisch", xlab=" ", las=1,
      ylab=" ", ylim=c(0,20))
lines(x, y2)
text( 7, 4, expression(y[1] == 1 + frac(10,x)), cex=1.3)
text( 5,16, expression(y[2] == 20 - frac(10,x)), cex=1.3)

x <- seq(0.1, 12, by=0.1); e <- exp(1)
y1 <- 0.1 * e^(0.5*x); y2 <-  10 * e^(-0.5*x)
plot(x, y1, type="l", main="exponentiell", xlab=" ", las=1,
      ylab=" ", ylim=c(0,20))
lines(x, y2)
text(7.7, 15, expression(y[1] == 0.1 * plain(e)^{0.5*x}), cex=1.3)
text(3,  8, expression(y[2] ==  10 * plain(e)^{-0.5*x}), cex=1.3)
```



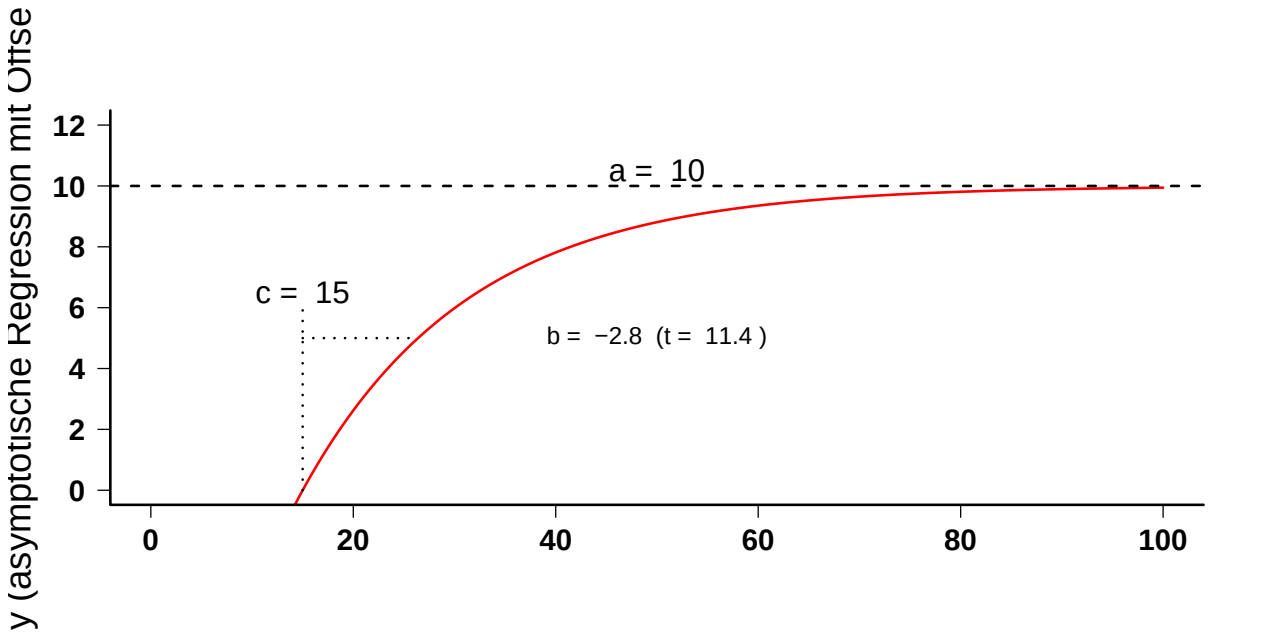
```r
x <- c(1, 2, 3, 4, 5)                                  # nichtlineare Regression
y <- c(4, 1, 3, 5, 6)
nls( y ~ a + b*x + c*x^2, start = list(a = 1, b = 1, c = 1))
## Nonlinear regression model
##   model: y ~ a + b * x + c * x^2
##    data: parent.frame()
##       a       b       c
##  5.4000 -2.6286  0.5714
##  residual sum-of-squares: 3.829
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 9.257e-09

                                                       # Güte der Anpassung
mod    <- nls( y ~ a + b*x + c*x^2, start = list(a = 1, b = 1, c = 1))
formula(mod); coef(mod)
## y ~ a + b * x + c * x^2
##          a          b          c
##  5.4000000 -2.6285714  0.5714286
new <- data.frame(x = c(1,2,3,4,5))
predict(mod, new)
## [1] 3.342857 2.428571 2.657143 4.028571 6.542857

a <- coef(mod) [1]; b <- coef(mod) [2]; c <- coef(mod) [3]
y.hat  <- predict(mod, new)
x.line <- seq(1, 5, by=0.1)
y.line <- a + b*x.line + c*x.line^2

par(mfrow=c(1,1), lwd=1.5, font.axis=2, bty="l", ps=12,
    cex.lab=1.4, cex.axis=1.2)
plot(x, y, xlab="x - Wert", ylab="y - Wert", las=1,
     ylim=c(1,7), cex=2, pch=16)
points(x, y.hat, cex=2, pch=18, col="red")
lines(x.line, y.line, lty=2, lwd=2, col="red")
```


y − Wert
x − Wert
1
2
3
4
5
6
7

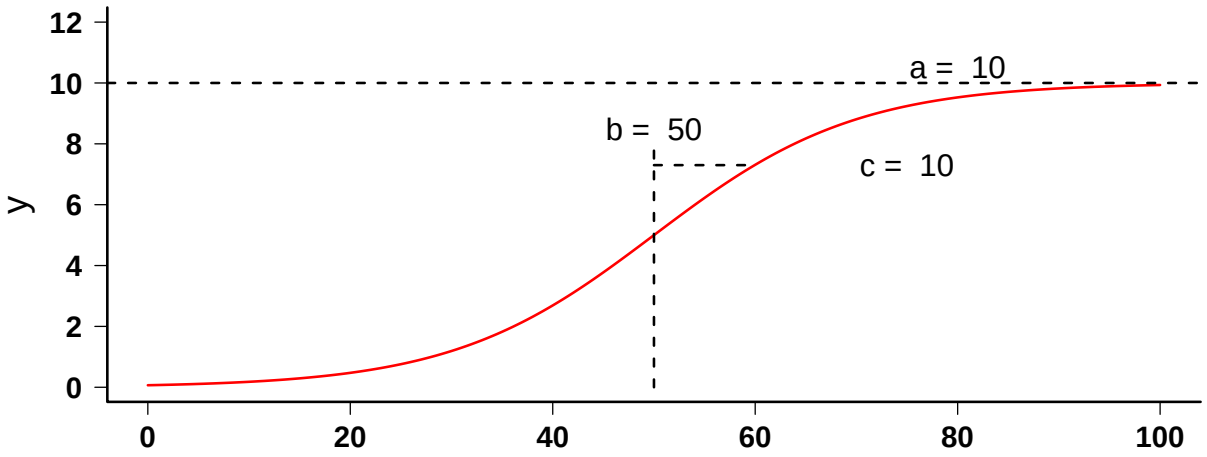
```r
x <- seq(0, 100,by=.5)                                  # asymptotische Regression
a <- 10; b <- 2 ; c <- -2.8
t0.5 <- log(2)/exp(c); t0.5
## [1] 11.39856
y.asymp <- a + (b-a)*exp(-exp(c)*x)
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14,
    cex.lab=1.5, cex.axis=1.2)
plot(x, y.asymp, type="l", las=1, col="red",
     ylim=c(0,12), xlab=" ", ylab="y")
abline(h=a, lty=2); abline(h=b, lty=2)
y1 <- a + (b-a)*exp(-exp(c)*t0.5)
lines(c(t0.5,t0.5), c(0,y1), lty=3)
lines(c(0, t0.5), c(y1, y1), lty=3)
text(50, a+0.5, paste("a = ",a), cex=1.3)
text(50, b+0.5, paste("b = ",b), cex=1.3)
text(45, b+(a-b)/2, paste("c = ",c," (t = ",round(t0.5,2),")"), cex=1.3)
```



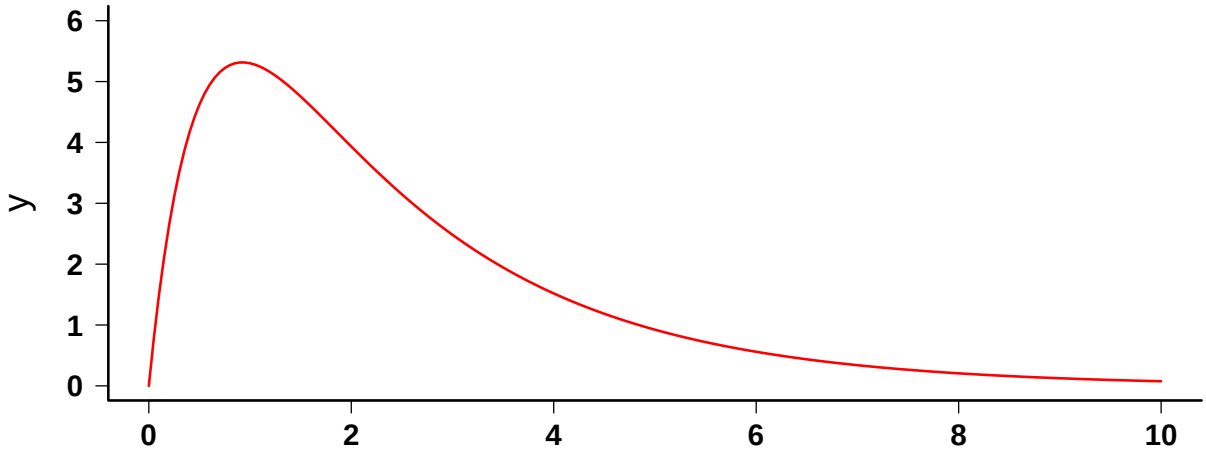
```r
x <- seq(0, 100,by=.5)                                  # asymptotische Regression mit Offset
a <- 10; b <- -2.8; c <- 15
t0.5 <- log(2)/exp(b); t0.5
## [1] 11.39856
y.asympoff <- a*(1-exp(-exp(b)*(x-c)))

par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14,
    cex.lab=1.5, cex.axis=1.2)
plot(x, y.asympoff, type="l", ylim=c(0,12), xlab=" ", las=1, col="red",
     ylab="y (asymptotische Regression mit Offset)")
abline(h=a, lty=2)
y1 <- a*(1-exp(-exp(b)*(x-c)))
lines(c(c, c), c(0, a/2+1), lty=3)
lines(c(c, c+t0.5), c(a/2, a/2), lty=3)
text(50, a+0.5, paste("a = ", a), cex=1.3)
text(c, a/2+1.5, paste("c = ",c), cex=1.3)
text(50, a/2, paste("b = ",b," (t = ",round(t0.5,2),")"))
```



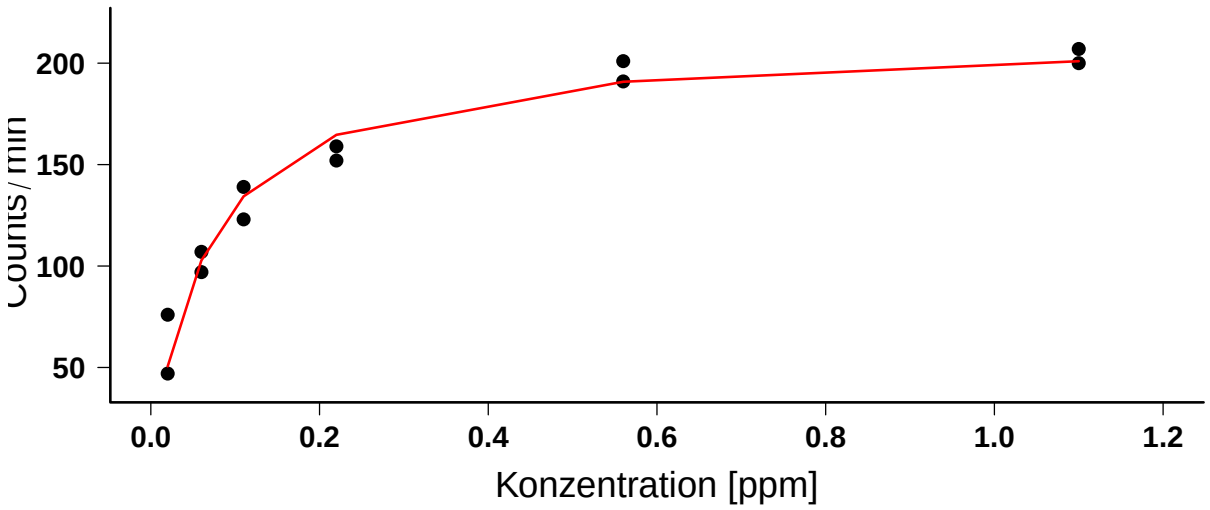
```r
x <- seq(0, 100,by=.5)                                  # logistische Regression
a <- 10; b <- 50;  c <- 10
y.logis <- a / (1 + exp((b-x)/c))
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14,
    cex.lab=1.5, cex.axis=1.2)
plot(x, y.logis, type="l", ylim=c(0,12), xlab=" ", ylab="y", las=1, col="red")
abline(h=a, lty=2)
y1 <- a - 2
lines(c(b, b), c(0, y1), lty=2)
y2 <- 0.73 * a; y2
## [1] 7.3
x2 <- b - c * log(1/0.73 - 1); x2
## [1] 59.94623
lines(c(b, x2), c(y2, y2), lty=2)
text(80, a+0.5, paste("a = ",a), cex=1.3)
text(50, y1 + 0.5, paste("b = ",b), cex=1.3)
text(75, y2, paste("c = ",c), cex=1.3)
```



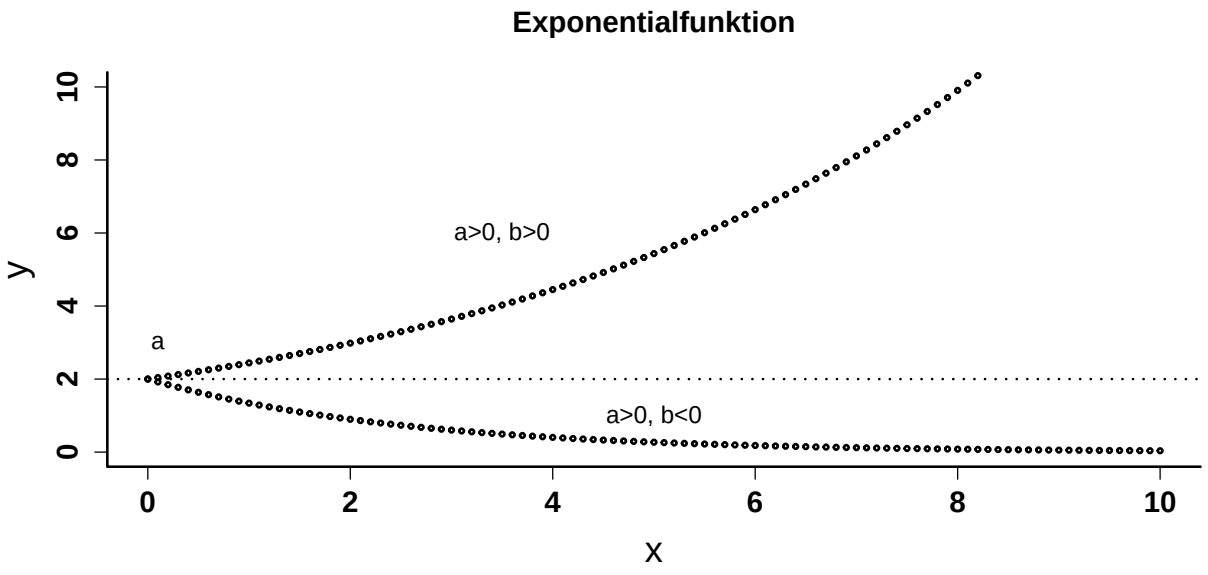
```r
x <- seq(0, 10, by=.05)                              # Compartment-Modell Regression
dose <- 1.5; elim <- 0.5; absorp <- 2.0; clear <- 0.2
a <- exp(elim); a
## [1] 1.648721
b <- exp(absorp); b
## [1] 7.389056
c <- exp(clear); c
## [1] 1.221403
y.fol <- (dose * elim * absorp / clear * (absorp-elim)) *
  (exp(-elim*x)-exp(-absorp*x))
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14,
    cex.lab=1.5, cex.axis=1.2)
plot(x, y.fol, type="l", ylim=c(0,6), xlab=" ", ylab="y", las=1, col="red")
```



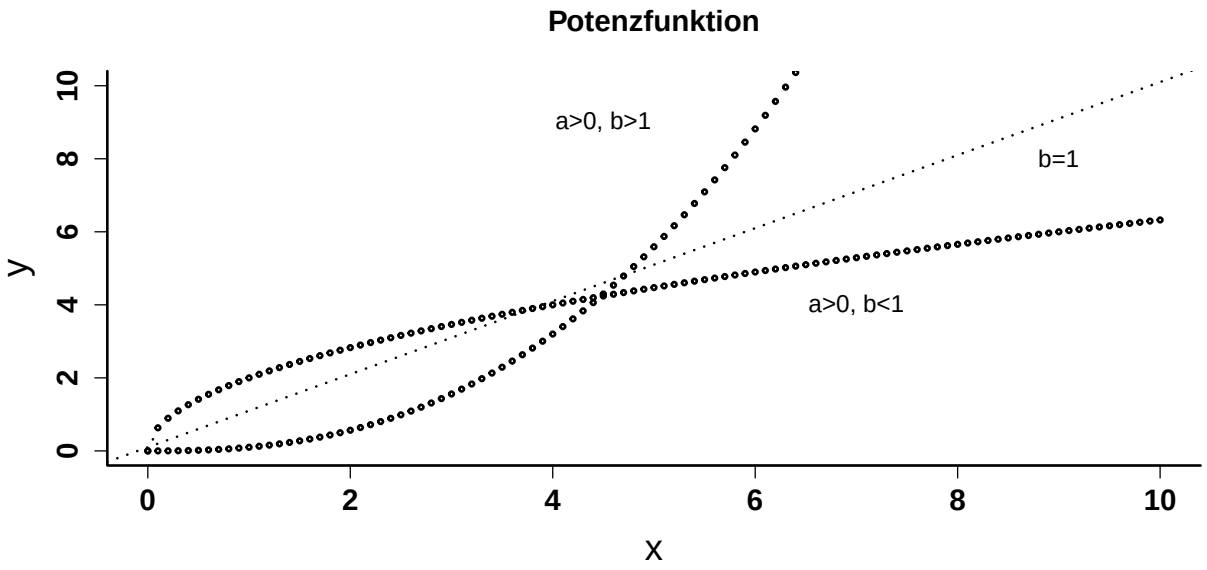
```r
                                                    # Michaelis-Menton Gleichung
conc <- c(0.02, 0.02, 0.06, 0.06, 0.11, 0.11, 0.22,
          0.22, 0.56, 0.56, 1.10, 1.10)
rate <- c(76,  47,  97, 107, 123, 139, 159, 152, 191, 201, 207, 200)
nls(rate ~ SSmicmen(conc, Vm, K))
## Nonlinear regression model
##   model: rate ~ SSmicmen(conc, Vm, K)
##    data: parent.frame()
##        Vm         K
## 212.68371   0.06412
##  residual sum-of-squares: 1195
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 1.937e-06
par(mfrow=c(1,1), lwd=2, font.axis=2, bty="l", ps=14,
    cex.lab=1.5, cex.axis=1.2)
fmod <- nls(rate ~ SSmicmen(conc, Vm, K))
plot(conc, rate, xlab="Konzentration [ppm]", las=1,
      ylab=expression(Counts / min^2),
     xlim=c(0,1.2), ylim=c(40,220), cex=1.5, pch=16)
lines(conc, predict(fmod, list(conc = conc)), lwd=2, col="red")
```



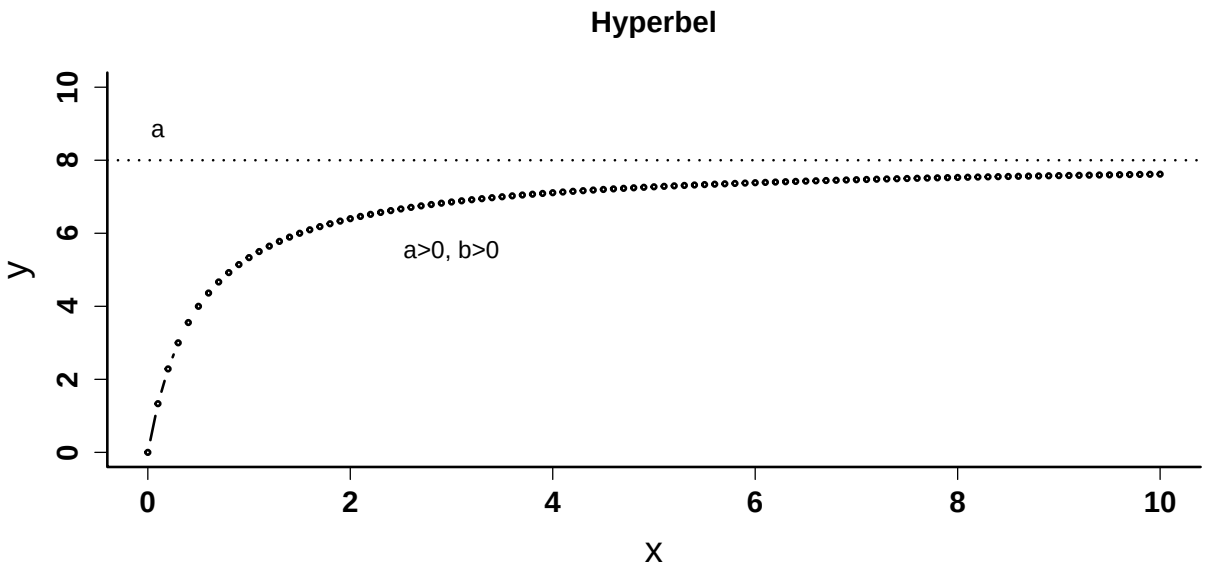
```r
x <- c(1, 2, 3, 4, 5)                                  # Beispiel zu den Normalgleichungen
y <- c(3, 7,12,26,51)
nls( y ~ a*b^x, start = list(a = 1, b = 1))
## Nonlinear regression model
##   model: y ~ a * b^x
##    data: parent.frame()
##      a     b
## 1.602 1.999
##  residual sum-of-squares: 1.225
##
## Number of iterations to convergence: 7
## Achieved convergence tolerance: 6.097e-06

                                                       # linearisierende Funktionen
x <- seq(0, 10, by=0.1)
a <- 2;  b <- 0.2
y1 <- a * exp(b*x)                                     # Exponentialfnktion
plot(x, y1, type="b", ylim=c(0, 10), cex = 0.5,
     ylab="y", main="Exponentialfunktion")
abline(h=2, lty=3)
text(0.1, 3, "a")
text(3.5, 6, "a>0, b>0")
a <- 2;  b <- -0.4
y1 <- a * exp(b*x)
lines(x, y1, type="b", ylim=c(0, 10), cex =0.5, lty = 2)
text(5, 1, "a>0, b<0")
```



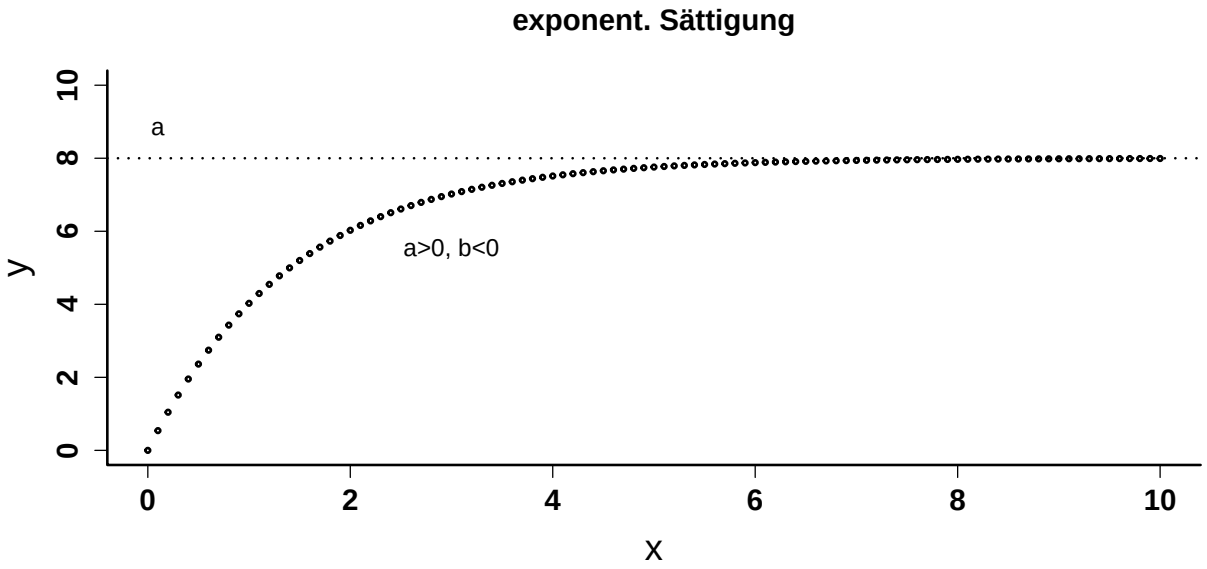
```r
a <- 0.1; b <- 2.5
y2 <- a * x^b                                    # Potenzfunktion
plot(x, y2, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="Potenzfunktion")
abline(a, 1, lty=3)
text(4.5, 9, "a>0, b>1")
text(9, 8, "b=1")
a <- 2; b <- 0.5
y2 <- a * x^b
lines(x, y2, type="b", ylim=c(0, 10), cex =0.5, lty = 2)
text(7, 4, "a>0, b<1")
```



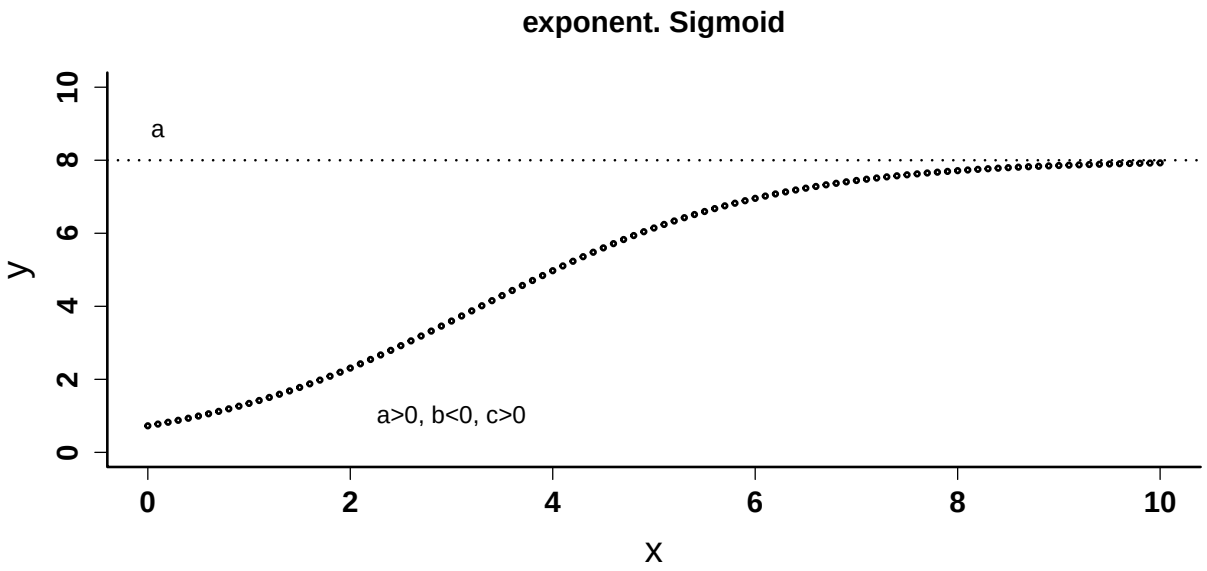
```
a <- 8; b <- 0.5
y3 <- a*x / (b + x)                                  # Hyperbelfunktion
plot(x, y3, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="Hyperbel")
abline(h=8, lty=3)
text(0.1, 8.8, "a")
text(3, 5.5, "a>0, b>0")
```



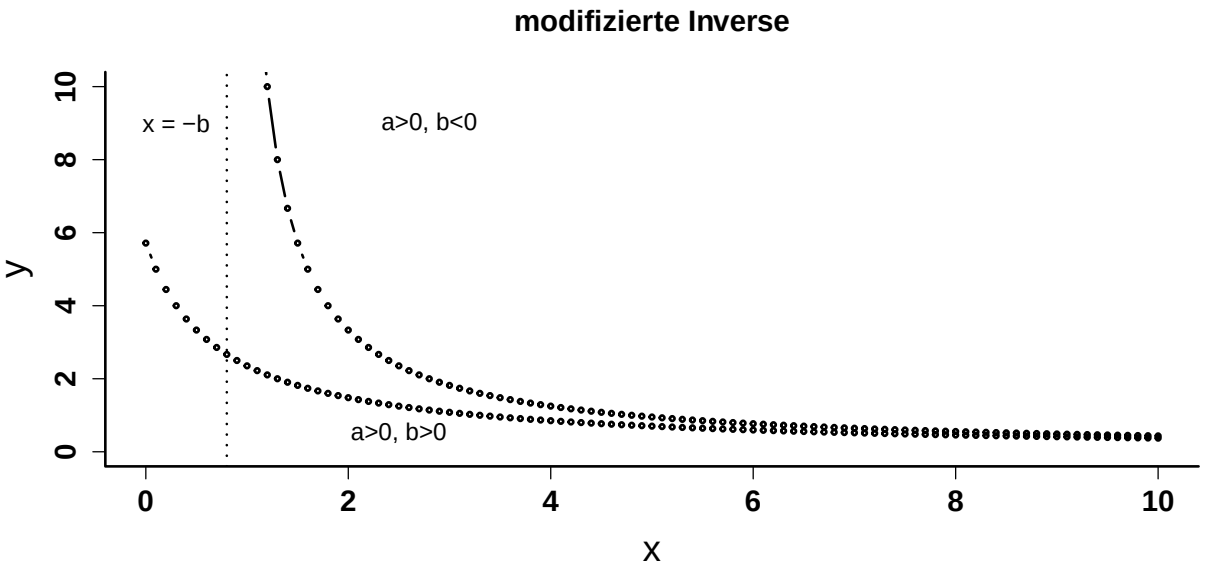
```
a <- 8; b <- -0.7
y4 <- (1 - exp(b*x))*a                              # exponent. Sättigung
plot(x, y4, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="exponent. Sättigung")
abline(h=8, lty=3)
text(0.1, 8.8, "a")
text(3, 5.5, "a>0, b<0")
```



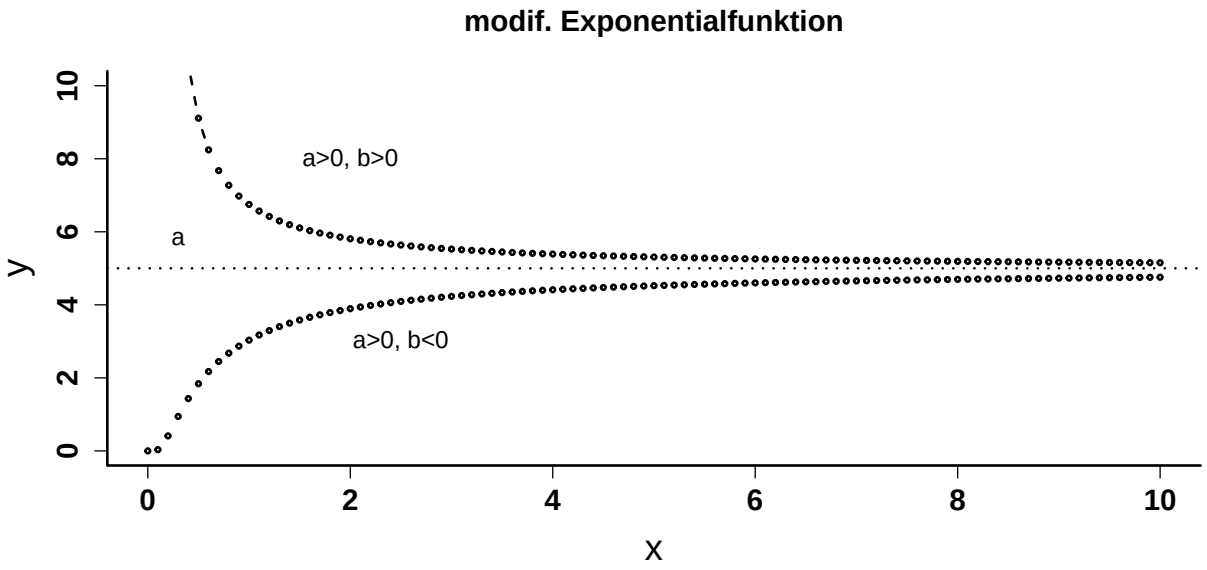
```r
a <- 8; b <- -0.7
y5 <- a / (1 + 10*exp(b*x))                          # exponent. Sigmoid
plot(x, y5, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="exponent. Sigmoid")
abline(h=8, lty=3)
text(0.1, 8.8, "a")
text(3, 1, "a>0, b<0, c>0")
```



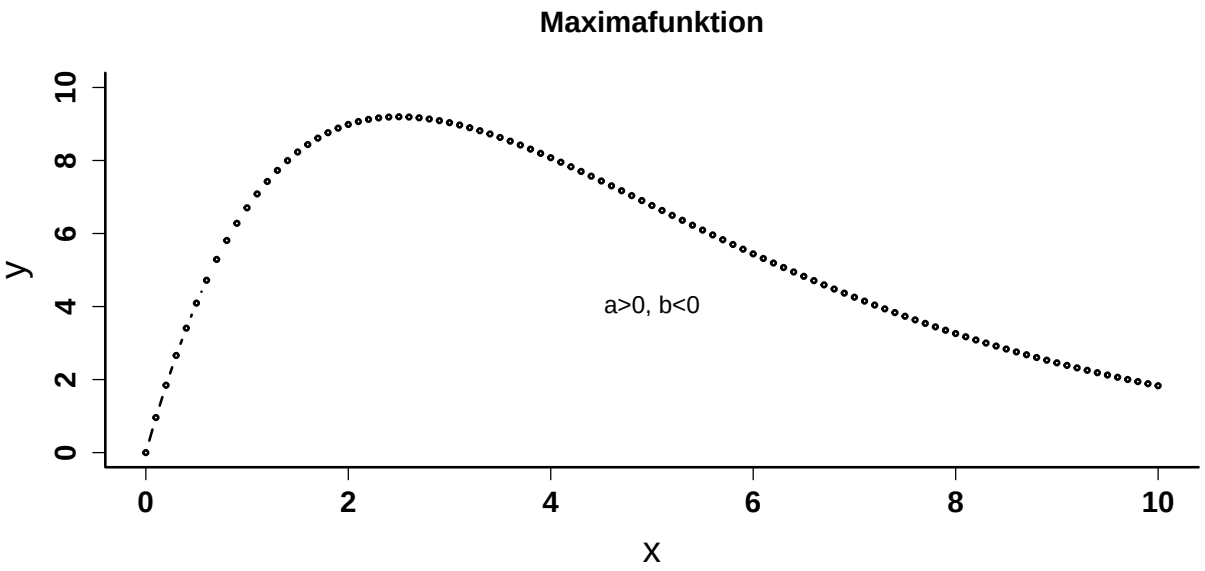
```r
a <- 4; b <- -0.8
y6 <- a / (b + x)                                  # modifizierte Inverse
plot(x, y6, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="modifizierte Inverse")
abline(h=a/b, lty=3)
abline(v=-b, lty=3)
text(0.3, 9, "x = -b")
text(2.8, 9, "a>0, b<0")
a <- 4; b <- 0.7
y6 <- a / (b + x)
lines(x, y6, type="b", ylim=c(0, 10), cex =0.5, lty = 2)
text(2.5, 0.5, "a>0, b>0")
```



```r
a <- 5; b <- -0.5
y7 <- a * exp(b/x)                                  # modifizierte Exponentialfunktion
plot(x, y7, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="modif. Exponentialfunktion")
abline(h=5, lty=3)
text(0.3,5.8,"a")
text(2.5, 3, "a>0, b<0")
a <- 5; b <- 0.3
y7 <- a * exp(b/x)
lines(x, y7, type="b", ylim=c(0, 10), cex =0.5, lty = 2)
text(2, 8, "a>0, b>0")
```



```r
a <- 10; b <- -0.4
y8 <- a * x * exp(b*x)                              # Maximafunktion
plot(x, y8, type="b", ylim=c(0, 10), cex =0.5,
     ylab="y", main="Maximafunktion")
text(5, 4, "a>0, b<0")
```



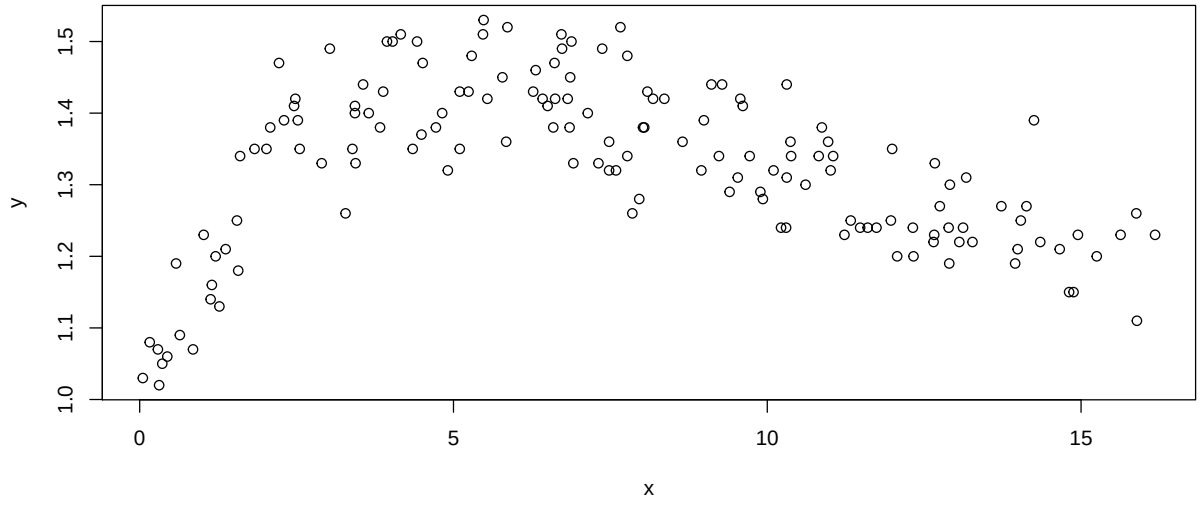
3.8 Nichtparametrische Regression

3.8.1 Regressogramm

```
dt <- read.csv2("nonparegdat.csv")
dt[1:10, ]
```

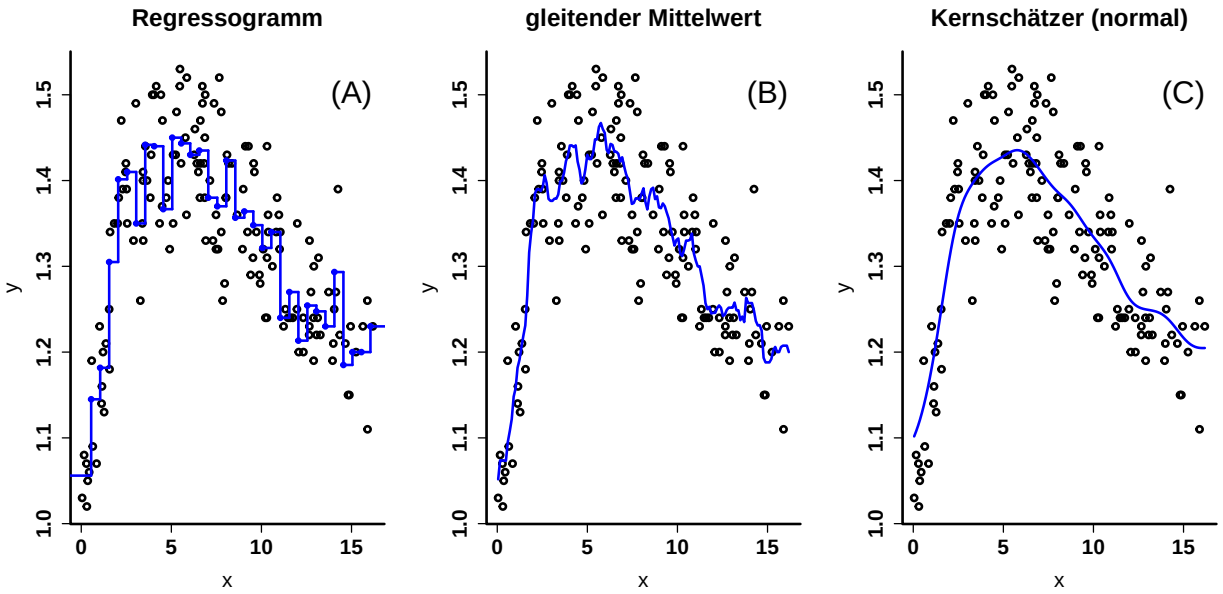
x	y
0.29	1.07
0.16	1.08
2.46	1.41
1.27	1.13
1.57	1.18
2.22	1.47
1.60	1.34
1.15	1.16
1.21	1.20
3.44	1.33

```
x <- dt$x; y <- dt$y

plot(x, y)
```



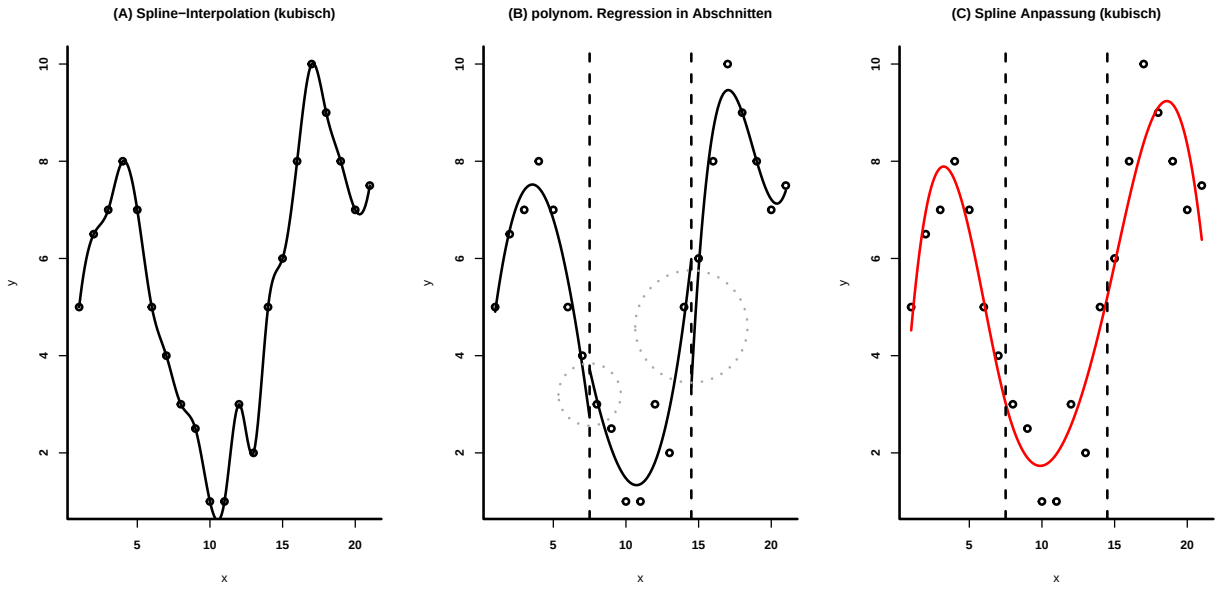

```
                                                    # Regressogramm
regramm <- function(x, y, h) {
  cuts <- seq(min(x), max(x)+h, by=h)
  bins <- cut(x, cuts)
  yval <- aggregate(y~bins, FUN=mean)
  return(stepfun(x=cuts[c(-1,-length(cuts))], y=yval$y))
```

```
}

par(mfcol=c(1,3), lwd=2, font.axis=2, bty="l", ps=18)
plot(x, y, main="Regressogramm")
lines(regramm(x, y, h=0.5), col="blue", lwd=2)
text(15,1.5,"(A)", cex=1.5)

plot(x, y, main="gleitender Mittelwert")                # gleitender Mittelwert
lines(ksmooth(x, y, kernel="box", bandwidth=1),  col="blue", lwd=2)
text(15,1.5,"(B)", cex=1.5)

plot(x, y, main="Kernschätzer (normal)")                # gleitende Anpassung
lines(ksmooth(x, y, kernel="normal", bandwidth=2),  col="blue", lwd=2)
text(15,1.5,"(C)", cex=1.5)
```



3.8.2 Kubische Spline Interpolation

```
x1 <- 1:7;   xs1 <- seq(1.0,   7.5, by=0.1)            # erzeugen künstlicher Daten
x2 <- 8:14;  xs2 <- seq(7.5,  14.5, by=0.1)
x3 <- 15:21; xs3 <- seq(14.5, 21.0, by=0.1)
x  <- c(x1, x2, x3); xs <- seq(1,21, by=0.1)
y1 <- c(5,6.5,7,8,7,5,4);
y2 <- c(3,2.5,1,1,3,2,5);
y3 <- c(6,8,10,9,8,7,7.5)
y  <- c(y1, y2, y3)
                                                       # 'truncate' Polynom 3. Grades
tpow <- function(x, t) (x - t) ^ 3 * (x > t)

par(mfrow=c(1,3), lwd=2, font.axis=2, bty="l", ps=10)
plot(x, y, main="(A) Spline-Interpolation (kubisch)")  # splines
lines(spline(x, y, xout=seq(1,21, by=0.1)))
plot(x, y, main="(B) polynom. Regression in Abschnitten")
abline(v=c(7.5, 14.5), lty=2)
s1 <- lm(y1 ~ x1 + I(x1^2) + I(x1^3))                          # 1. Abschnitt
lines(xs1, predict(s1, data.frame(x1=xs1)))
s2 <- lm(y2 ~ x2 + I(x2^2) + I(x2^3))                          # 2. Abschnitt
lines(xs2, predict(s2, data.frame(x2=xs2)))
s3 <- lm(y3 ~ x3 + I(x3^2) + I(x3^3))                          # 3. Abschnitt
lines(xs3, predict(s3, data.frame(x3=xs3)))
symbols(7.5, 3.2, circles = 0.2, inches=0.25, add=TRUE,
        lty=3, fg="darkgrey")
symbols(14.5, 4.6, circles = 0.2, inches=0.45, add=TRUE,
        lty=3, fg="darkgrey")
                                                       # Spline-Interpolation (kubisch)
plot(x, y, main="(C) Spline Anpassung (kubisch)")
fm <- lm(y ~ x + I(x^2) + I(x^3) + I(tpow(x,7.5)) + I(tpow(x,14.5)))
xs <- seq(1, 21, by=0.1)
lines(xs, predict(fm, data.frame(x=xs)), col="red")
abline(v=c(7.5, 14.5), lty=2)
```


(A) Spline−Interpolation (kubisch)
(B) polynom. Regression in Abschnitten
(C) Spline Anpassung (kubisch)
y
x
2
4
6
8
10
5
10
15
20