# Neues Labor: Basics für Module zur Datenanalyse

Peter Naeve

13.02.2003

# 1 danalyse.r

## 1.1 Zweck

Diese `library` stellt für die 4 Module zur Datenanalyse (Deskriptive Statistik) Funktionen bereit, die den Modulen angepaßt sind, d.h. sie "heilen die sonst im R-Kalkulator regierende R-spezifische Semantik.
Dieser Text ist hauptsächlich für Dozenten gedacht, damit sie

1. die Semantik der in den Modulen eingeführten Funktionen besser verstehen
2. einen Ansatz haben, um diese Funktionen gegebenfalls ihren spezifischen Wünschen anzupassen.

Aber natürlich kann er auch dem interessierten fortgeschritten Studenten als Einstieg in den Aspekt *Computational Statistics* geben.

## 1.2 Logik der Fallunterscheidung

Die Funktionen sollen auf einer oder mehreren der Eingaben Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle arbeiten. Die unterschiedlichen Fälle werden wie folgt erkannt. Die Dateneingabe ist immer mit `x` bezeichnet.

| | |
|---|---|
| Zulässige Eingabe | |
| Urliste | `is.vector(x) && is.numeric(x) == T` |
| Häufigkeitstabelle | `xclass <- class(x)` |
| | `(length(xclass>1 && xclass[2] == "Htab") == T` |
| diskrete Htab | `xclass <- class(x)` |
| | `(length(xclass>1 && xclass[1] == "dHtab") == T` |
| kontinuierliche Htab | `xclass <- class(x)` |
| | `(length(xclass>1 && xclass[1] == "kHtab") == T` |
| Typ der Eingabe bei zulässiger Eingabe | |
| Urliste | `is.vector(x) == T` |
| diskrete Htab | `xclass[1]=="dHtab" == T` |
| kontinuierliche Htab | `xclass[1]=="kHtab" == T` |

## 1.3 Funktionen der oberen Ebene

Hier folgt eine Auflistung der dem Leser in den 4 Modulen zur Datenanalyse begebnenden Funktionen.

Alle Funktionen arbeiten in zwei Modi, lautlos oder mit Informationen zu Fallunterscheidungen und Fehlersituationen. Die Defaulteinstellung ist “lautlos.

## 1.4 Funktionen der unteren Ebene

Diese Funktionen sind die Knechte in der Finsternis. Nur der Veränderer dieser Bibliothek muß] sich wirklich mit diesen Funktionen befassen. Alle anderen verlieren fast nichts, wenn sie die einschl”agigen Textpassagen überspringen.

| | |
|---|---|
| baw | für box.and.whisker |
| dHtab | Bildet Klasse diskrete Häufigkeitstabelle |
| kHtab | Bildet Klasse kontinuierliche Häufigkeitstabelle |
| nformat | für five.values |
| print.Htab | Methode für Htab |
| print.dHtab | Methode für dHtab |
| print.kHtab | Methode für kHtab |
| xo.nice | Berechnet schöne Obergrenze |
| xu.nice | Berechnet schöne Untergrenze |

## 1.5 Library Description

Gemäß der Struktur des Statistiklabors benötigt jede Bibliothek eine sogenannte `library description`. Diese wird jetzt bereitgestellt.
!!! Erste Zeile: quick und dirty !!!

1 ⟨*start* 1⟩≡

```
#0:
##LibName#danalyse.r
##LibDescription#Eine Sammlung von Funktionen zur Datenanalyse
##LibDescription#Zur Zeit noch Teststadium
##LibDescription#version vom 04.02.03
##LibCopyright#Peter Naeve
```

This definition is continued in chunks 3, 9, 13, 18, 21, 25, 29, 30, 34–36, 38, 44, 50, 61, 63, 67, 73, 84, 92, 100, 108, 116, 124, and 131.

Root chunk (not used in this document).

## 1.6 Produktion

Die Library wird mit Hilfe des REVWEB-Systems erstellt. Aus dem von diesem erzeugten `.sch-` bzw, `.r`-File sind jeweils nur die Zeilen mit `#0:` — bei einigen REVWEB-versionen wird sie gar nicht erst erzeugt — und tt #:0, sowie die die nach der Zeile mit dem Beginn `#:0` folgenden Testteile zu entfernen. Ebenfalls sind die Zeilen mit dem Code `SAVE.r ...` auszukommentieren oder zu entfernen. Gegebenenfalls ist die Extension auf `.r` zu setzen.

# 2 Exploratives von Tukey und Co

## 2.1 Fünf-Zahlen-Zusammenfassung

Steckbrief siehe Seite 59
Die *Fünf-Zahlen-Zusammenfassung* wird in zwei Versionen angeboten. Zum einen die simple Form aus dem Text "Neue Statistik" und zum anderen die von Tukey in seinem Buch *Explorative Data Analysis* vorgestellte Fassung.

### Labor Information

2 ⟨*five.values.Labor* 2⟩≡
```
##FunctionName#five.values
##FunctionDescription#Fuenf-Zahlen-Zusammenfassung
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor, Urliste oder Rangwertreihe
##FunctionDescription#   NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#tukey: ==F einfache Version  (Default)
##FunctionDescription#       ==T Tukey-Stil
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#Fuenf-Zahlen-Darstellung als string
##FunctionDescription# last change 27.01.03
##FunctionSyntax#five.values(x), five.value(x,,F), five,values(x,T), ...
```
This code is used in chunk 3.

### Struktur five.values

3 ⟨*start* 1⟩+≡
```
five.values <- function(x,tukey=F,SIL=T) {
⟨five.values.Labor 2⟩
⟨five.values.init 4⟩
if (is.vector(x) && is.numeric(x)) {
⟨five.values.Berechnung.string 5⟩
⟨five.values.erg.tukey 6⟩
⟨five.values.erg.simple 7⟩
}else{
  if(SIL == F) cat("\nfive.values: Keine zulaessige Eingabe!\n") }
return(erg)
invisible()
}
```

### Refinements five.values

*five.values.init*

4 ⟨*five.values.init* 4⟩≡

```
erg <- NA
```

This code is used in chunk 3.

*five.values.Berechnung.string*

5 ⟨*five.values.Berechnung.string* 5⟩≡

```
xh  <- x[!is.na(x)]
xh  <- sort(xh)
n   <- length(xh)
h   <- c(" ", " ", " ", " ")
i   <- ceiling(n/2)
d   <- c(n, i)
j   <- ceiling((1 + n)/2)
w   <- (xh[i] + xh[j])/2
if(!(i == j)) h[2] <- "h"
k   <- i
i   <- ceiling(k/2)
j   <- ceiling((1 + k)/2)
d   <- c(d, i, 1)
if(!(i == j)) h[3] <- "h"
w   <- c(w,((xh[i] + xh[j])/2),((xh[n-i+1] + xh[n-j+1])/2),xh[1],xh[n])
d   <- format(d)
w   <- format(w)
k   <- nformat(w[1])
uk  <- rep("_", k)
bk  <- rep(" ", k)
nl  <- "\n"
```

This code is used in chunk 3.

*five.values.erg.tukey*

6 ⟨*five.values.erg.tukey* 6⟩≡

```
if (tukey==T) {
mw0 <- c(" # ", d[1], h[1], "   ", uk, uk, uk, nl)
mw1 <- c(" M ", d[2], h[2], " | ", bk, w[1], bk, " | ", nl)
mw2 <- c(" H ", d[3], h[3], " | ", w[2], bk, w[3], " | ", nl)
mw3 <- c("   ", d[4], h[4], " | ", w[4], bk, w[5], " | ", nl)
erg <- cat(mw0, mw1, mw2, mw3, sep = "") }
```

This code is used in chunk 3.

*five.values.erg.simple*

7 ⟨*five.values.erg.simple* 7⟩≡

```
if (tukey==F) {
mw0 <- c("   ", uk, uk, uk, nl)
mw1 <- c(" | ", bk, w[1], bk, " | ", nl)
mw2 <- c(" | ", w[2], bk, w[3], " | ", nl)
mw3 <- c(" | ", w[4], bk, w[5], " | ", nl)
erg <- cat(mw0, mw1, mw2, mw3, sep = "") }
```

This code is used in chunk 3.

### 2.1.1 nformat

Wird von `five.values` benötigt.

**Labor Information**

8 ⟨*nformat.Labor* 8⟩≡

```
##FunctionName#nformat
##FunctionDescription#wird in five.values benoetigt
##FunctionDescription#Parameter:~
##FunctionDescription#string: ein String
##FunctionDescription#Value:~
##FunctionDescription#Formatlaenge
##FunctionDescription# last change 14.09.02
##FunctionSyntax#nformat(string)
```

This code is used in chunk 9.

### Struktur nformat

9 ⟨*start* 1⟩+≡

```
nformat <- function(string) {
⟨nformat.Labor 8⟩
st <- (format(c("", string)))[1]
z  <- list(" ","  ","   ","    ","     ","      ","       ",
           "        ","         ","          ","           ")
k  <- 0
j  <- 1
for(i in z) { if(i == st) k <- j
              j <- j + 1 }
return(k)
invisible()
}
SAVE.r("nformat")
```

### 2.1.2 Test: five.values

10 ⟨* 10⟩≡

```
cat("\nSimpler Stil\n")
print(five.values(1:40))
```

This definition is continued in chunks 11, 19, 22, 23, 26, 42, 48, 64, 65, 71, 80–82, 90, 98, 106, 114, 122, and 130.

Root chunk (not used in this document).

11 ⟨* 10⟩+≡

```
cat("\nTukey Stil\n")
print(five.values(1:40,T))
```

## 2.2 box.and.whisker

Steckbrief siehe Seite 59
Es sollen ein oder mehrere Box-and-Whisker-Plots in einer Graphik vereint werden. Dazu muß der Benutzer die einzelnen Datensätze mitgeben.
Auf hoher Ebene ist die Funktion rasch skizziert.

### Labor Information

12 ⟨*box.and.whisker.Labor* 12⟩≡

```
##FunctionName#box.and.whisker
##FunctionDescription#Ein oder mehrere Box-and-Whisker-Plot in einer Graphik
##FunctionDescription#Parameter:~
##FunctionDescription#... liste der Datensaetze
##FunctionDescription#kenn: Vektor der Kennungen der Datensaetze, (optional)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#box.and.whisker(...,kenn)
```

This code is used in chunk 13.

### Struktur box.and.whisker

13 ⟨*start* 1⟩+≡

```
box.and.whisker <- function(..., kenn=""){
⟨box.and.whisker.Labor 12⟩
⟨box.and.whisker.Init 14⟩
⟨box.and.whisker.plot.frame 15⟩
⟨box.and.whisker.plot.baw 16⟩
invisible() }
```

**Refinements box.and.whisker**

*box.and.whisker.Init*

Verfeinern wir der Reihe nach. Für die Dimensionierung der Graphik ist es notwendig, das Minimum der Minima und das Maximum der Maxima zu erhalten. Zuvor muß man sich einen Überblick über die Anzahl der zu zeichnenden Box-and-Whisker-Plots verschaffen.

14 ⟨*box.and.whisker.Init* 14⟩≡

```
n <- nargs()
if(kenn[1] != "") n <- n-1
all.x <- list(...)
xmin  <- numeric(n)
xmax  <- numeric(n)
for (i in 1:n){
  x       <- all.x[[i]]
  xmin[i] <- min(x)
  xmax[i] <- max(x) }
xmin  <- min(xmin)
xmax  <- max(xmax)
```

This code is used in chunk 13.

*box.and.whisker.plot.frame*

Die Initialisierung des Plots erfordert eine Entscheidung über die Lage der Plots. Sie sollen horizontal angeordnet werden, d.h. die Merkmalsachse ist die x-Achse. Diese soll im Bereich der Werte gezeichnet werden. In vertikaler Richtung soll keine Achse gezeichnet werden.

15 ⟨*box.and.whisker.plot.frame* 15⟩≡

```
#setze Benutzer-Koordinatenbereich
y  <- 1.0
dy <- 0.5
xp <- c(xmin,xmax)
yp <- c(0,(2*y*n))
plot(xp,yp,type="n",axes=F,xlab="",ylab="")
title("Box and Whisker Plot")
par(xaxp=c(xmin,xmax,5))
axis(1,ticks=T,pos=0.2)
```

This code is used in chunk 13.

*box.and.whisker.plot.baw*

Die Aufbereitung der Daten bedeutet die Bereitstellung der fünf Werte (Extremwerte, Angeln und Median). Der jeweilige Plot ist Dank des Templates rasch getan.

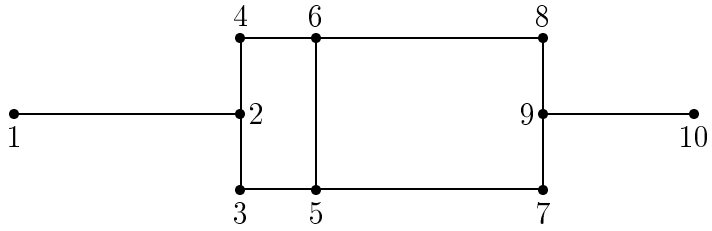
16 ⟨*box.and.whisker.plot.baw* 16⟩≡

```
for (i in 1:n){
x <- all.x[[i]]
x <- quantile(x)
baw(x,y,dy)
if(kenn[1] == "")
  tt <- paste("Box.and.Whisker ",format(i),sep="")
else
  tt <- paste("Box.and.Whisker ",kenn[i],sep="")
text(xmin,y+1,label=tt)
# next baw
y <- y+2 }
```

This code is used in chunk 13.

### 2.2.1 baw

Wird von box.and.whisker benötigt. Erstellt einen nackten Box.and.Whisker. Es wird eine R-Funktion entwickelt werden, die die allgemeine Form der nachstehenden Figur erzeugt, so daß die Figur an beliebiger Stelle zu einer Graphik hinzugefügt werden kann.




Die Figur ist durch die 10 gekennzeichneten Punkte beschrieben. Die x-Koordinaten der Punkte sind dabei den darzustellenden statistischen Größen — sie werden als Vektor x übergeben — wie folgt zugeordnet.

| | | |
|---|---|---|
| x[1] | $x_{min}$ | 1 |
| x[2] | untere Angel | 2, 3, 4 |
| x[3] | $\tilde{x}$ | 5, 6 |
| x[4] | obere Angel | 7, 8, 9 |
| x[5] | $x_{max}$ | 10 |

Die y-Koordinaten sind y, y+dy und y-dy, mit den Zuordnungen

| | |
|---|---|
| y+dy | 4, 6, 8 |
| y | 1, 2, 9, 10 |
| y-dy | 3, 5, 7 |

Damit liegt die nachstehende Funktion fast auf der Hand.

#### Labor Information

17 ⟨*baw.Labor* 17⟩≡

```
##FunctionName#baw
##FunctionDescription#wird in box.and.whisker benoetigt
##FunctionDescription#Parameter:~
##FunctionDescription#x: x-Koordinaten box and whisker
##FunctionDescription#y: y-Koordinaten box and whisker
##FunctionDescription#dy: Boxbreite
##FunctionDescription# last change 15.09.02
##FunctionSyntax#baw(x,y,dy)
```

This code is used in chunk 18.

**Struktur baw**

18 ⟨*start* 1⟩+≡

```
baw<-function(x,y,dy){
⟨baw.Labor 17⟩
yu<-y-dy
yo<-y+dy
lines(c(x[1],x[2]),c(y,y),type="l")
lines(c(x[2],x[2]),c(yu,yo),type="l")
lines(c(x[3],x[3]),c(yu,yo),type="l")
lines(c(x[4],x[4]),c(yu,yo),type="l")
lines(c(x[2],x[4]),c(yu,yu),type="l")
lines(c(x[2],x[4]),c(yo,yo),type="l")
lines(c(x[4],x[5]),c(y,y),type="l") }
SAVE.r("baw")
```

### 2.2.2 Test box.and.whisker

19 ⟨* 10⟩+≡

```
x1 <- 1:20
x2 <- 5:25
cat("\nTest box.and.whisker\n")
cat(\"box.and.whisker(x1,x2)\n")
box.and.whisker(x1,x2)
```

## 2.3 upper.versus.lower

Steckbrief siehe Seite 60
Graph des oberen Teils eines Datensatze gegen den unteren Teil. Zur Information wird die bei Symmetrie zu erwartende Gerade in die Punktewolke gezeichnet.

### Labor Information

20 ⟨*upper.versus.lower.Labor* 20⟩≡

```
##FunctionName#upper.versus.lower
##FunctionDescription#Graph "Obere gegen Untere Datenhaelfte"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor, Urliste oder Rangwertreihe
##FunctionDescription#   NA's zugelassen, werden vor Zeichnung entfernt
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#upper.versus.lower(x), upper.versus.lower(x,F)
```

This code is used in chunk 21.

### Struktur upper.versus.lower

21 ⟨*start* 1⟩+≡

```
upper.versus.lower <- function(x,SIL=T){
⟨upper.versus.lower.Labor 20⟩
if (is.numeric(x) && is.vector(x)) {
  xh <- x[!is.na(x)]
  h  <- length(x)-length(xh)
  if(SIL == F && h > 0) {
  cat(paste("\nupper.versus.lower: Es wurden", h, "NAs entfernt\n"))}
  n  <- length(xh)
  xs <- sort(xh)
  nh <- floor(n/2)
  plot(xs[1:nh],xs[(n+1)-(1:nh)],xlab="lower",ylab="upper")
  abline(2*median(xh),-1)
  title("Upper versus Lower Plot")
}else{
  if(SIL == F) cat("\nupper.versus.lower: Keine zulaessige Eingabe!\n") }
invisible() }
```

### 2.3.1 Test upper.versus.lower

22 ⟨* 10⟩+≡

```
cat("\nTest upper.versus.lower\n")
cat("\nupper.versus.lower(1:10)\n")
print(upper.versus.lower(1:10))
```

23 ⟨* 10⟩+≡

```
cat("\nupper.versus.lower(xtestb)\n")
print(upper.versus.lower(xtestb))
cat("\nupper.versus.lower(c(NA,1:10))\n")
print(upper.versus.lower(c(NA,1:10)))
```

## 2.4 Spannweite

Steckbrief siehe Seite 65
Die Funktion `Spannweite` tut das,was ihr Name verspricht.

### Labor Information

24 ⟨*Spannweite.Labor* 24⟩≡

```
##FunctionName#Spannweite
##FunctionDescription#"range" auf deutsch
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor: Urliste oder Rangwertreihe
##FunctionDescription#   NA‘s zugelassen, werden vor Berechnung entfernt
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#max(x) - min(x) oder NA
##FunctionDescription# last change 27.01.03
##FunctionSyntax#Spannweite(x), Spannweite(x,F)
```

This code is used in chunk 25.

### Struktur Spannweite

25 ⟨*start* 1⟩+≡

```
Spannweite <- function(x,SIL=T) {
⟨Spannweite.Labor 24⟩
erg <- NA
if (is.numeric(x) && is.vector(x)) {
  xh <- x[!is.na(x)]
  h  <- length(x)-length(xh)
  if(SIL == F && h > 0) {
  cat(paste("\nSpannweite: Es wurden", h, "NAs entfernt\n"))}
  xh  <- range(xh)
  erg <- xh[2]-xh[1]
}else{
  if(SIL == F) cat("\nSpannweit: Keine zulaessige Eingabe!\n")}
return(erg)
invisible() }
```

### 2.4.1 Test Spannweite

26 ⟨* 10⟩+≡

```
cat("\nTest Spannweite\n")
cat("\nSpannweite(1:10)\n")
print(Spannweite(1:10))
cat("\nSpannweite(c(NA,1:10))\n")
print(Spannweite(c(NA,1:10)))
```

# 3 Klasse: Häufigkeitstabelle

## 3.1 Klassendefinition

Es werden zwei Klassen betrachtet:

| | |
|---|---|
| "dHtab" | Häufigkeitstabelle diskrete |
| "kHtab" | Häufigkeitstabelle kontinuierlich |

In Bezug auf das Drucken werden beide als Klasse "Htab" behandelt.

### Labor Information

27 ⟨*dHtab.Labor* 27⟩≡

```
##FunctionName#dHtab
##FunctionDescription#Erzeugt Klasse "dHtab"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Objekt, das Klasse bekommen soll
##FunctionDescription#Value:~
##FunctionDescription#x: mit Attribut class = "dHtab"
##FunctionDescription# last change 24.09.02
##FunctionSyntax#dHtab(x)
```

This code is used in chunk 29.

28 ⟨*kHtab.Labor* 28⟩≡

```
##FunctionName#kHtab
##FunctionDescription#Erzeugt Klasse "kHtab"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Objekt, das Klasse bekommen soll
##FunctionDescription#Value:~
##FunctionDescription#x: mit Attribut class = "kHtab"
##FunctionDescription# last change 24.09.02
##FunctionSyntax#kHtab(x)
```

This code is used in chunk 30.

### Stuktur dHtab, kHtab

29 ⟨*start* 1⟩+≡

```
dHtab <- function(x) {
⟨dHtab.Labor 27⟩
y <- x
class(y) <- c("dHtab", "Htab")
y }
SAVE.r("dHtab")
```

30 ⟨*start* 1⟩+≡

```
kHtab <- function(x) {
⟨kHtab.Labor 28⟩
y <- x
class(y) <- c("kHtab", "Htab")
y }
SAVE.r("kHtab")
```

## 3.2 print Routinen

### 3.2.1 `print.Htab`

Die eigntliche `print`-Funktion. Die Funktionen `print.dHtab` und `print.kHtab` sind *auf Vorrat*, es könnte ja sein, daß man eine unterschiedliche Behandlung anstrbt. Wir fallen auf den Typ Matrix zurück, indem die Klasse gelöscht wird.

**Labor Information**

31 ⟨*print.Htab.Labor* 31⟩≡

```
##FunctionName#print.Htab
##FunctionDescription#Methode fuer Klasse "Htab"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Objekt, das geprintet werden soll
##FunctionDescription# last change 24.09.02
##FunctionSyntax#print.Htab(x)
```

This code is used in chunk 34.

32 ⟨*print.dHtab.Labor* 32⟩≡

```
##FunctionName#print.dHtab
##FunctionDescription#Methode fuer Klasse "dHtab"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Objekt, das geprintet werden soll
##FunctionDescription# last change 24.09.02
##FunctionSyntax#print.dHtab(x)
```

This code is used in chunk 35.

33 ⟨*print.kHtab.Labor* 33⟩≡

```
##FunctionName#print.kHtab
##FunctionDescription#Methode fuer Klasse "kHtab"
##FunctionDescription#Parameter:~
##FunctionDescription#x: Objekt, das geprintet werden soll
##FunctionDescription# last change 24.09.02
##FunctionSyntax#print.kHtab(x)
```

This code is used in chunk 36.

**Struktur print-Methoden**

34 ⟨*start* 1⟩+≡

```
print.Htab <- function(x) {
⟨print.Htab.Labor 31⟩
class(x) <- NULL
NextMethod("print")
invisible() }
```

### 3.2.2 `print.dHtab`

Die Klasse wird auf `Htab` gestellt.

35 ⟨*start* 1⟩+≡

```
print.dHtab <- function(x) {
⟨print.dHtab.Labor 32⟩
class(x) <- "Htab"
NextMethod("print.Htab")
invisible() }
```

### 3.2.3 `print.kHtab`

Die Klasse wird auf `Htab` gestellt.

36 ⟨*start* 1⟩+≡

```
print.kHtab <- function(x) {
⟨print.kHtab.Labor 33⟩
class(x) <- "Htab"
NextMethod("print.Htab")
invisible() }
```

# 4 Diskrete Häufigkeitsverteilung

## 4.1 Häufigkeitstabelle

Steckbrief siehe Seite 61
Erstellt eine Häufigkeitstabelle für den diskreten Fall aus einer Urliste. Andere Eingabe wird zurückgewiesen. NAs werden entfernt.
Die diskrete Häufigkeitstabelle `x` hat folgende Struktur.

| | |
|---|---|
| `x[,1]` | $x_i$ |
| `x[,2]` | $n_i$ |
| `x[,3]` | $h_i$ |

### Labor Information

37 ⟨*DiskHaeuf.Labor* 37⟩≡

```
##FunctionName#DiskHaeuf
##FunctionDescription#Berechnet diskrete Haeufigkeitstabelle
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor, Urliste oder Rangwertreihe
##FunctionDescription#   NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#DiskHaeuf(x), DiskHaeuf(x,F)
```

This code is used in chunk 38.

### Struktur DiskHaeuf

38 ⟨*start* 1⟩+≡

```
DiskHaeuf <- function(x,SIL=T) {
⟨DiskHaeuf.Labor 37⟩
⟨DiskHaeuf.Init 39⟩
if (GO) {
  ⟨DiskHaeuf.Urliste 40⟩
}else{
  ⟨DiskHaeuf.Error 41⟩
     }
return(erg)
invisible() }
```

**Refinements DiskHaeuf**

*DiskHaeuf.Init*

39 ⟨*DiskHaeuf.Init* 39⟩≡

```
erg <- NA
GO <- F
if (is.vector(x) && is.numeric(x)) GO <- T
```

This code is used in chunk 38.

*DiskHaeuf.Urliste*

40 ⟨*DiskHaeuf.Urliste* 40⟩≡

```
# drop NAs
xh    <- x[!is.na(x)]
h     <- length(x) -length(xh)
if (SIL == F && h > 0) {
cat(paste("\nDiskHaeuf: Es wurden",h,"NAs entfernt\n"))}
n     <- length(xh)
abs.h <- table(xh)
xh    <- as.numeric(dimnames(abs.h)[[1]])
rel.h <- abs.h/n
rel.h <- zapsmall(rel.h)
erg   <- cbind(xh, abs.h, rel.h)
# falls man etwas anderes will
null  <- NULL
dimnames(erg) <- list(null, c("xi", "ni", "ni/n"))
erg   <- dHtab(erg)
```

This code is used in chunk 38.

*DiskHaeuf.Error*

41 ⟨*DiskHaeuf.Error* 41⟩≡

```
if(SIL == F) cat("\nDiskHaeuf: Kein zulaessiger Datensatz!\n")
```

This code is used in chunk 38.

### 4.1.1 Test DiskHaeuf

42 ⟨* 10⟩+≡

```
cat("\nErzeugung diskreter Haeufigkeitstabellen\n")
cat("\nDiskHaeuf(xtestb)\n")
print(DiskHaeuf(xtestb))
cat("\nDiskHaeuf(xtestd)\n")
print(DiskHaeuf(xtestd))
cat("\nDiskHaeuf(c(NA,NA,xtestb))\n")
print(DiskHaeuf(c(NA,NA,xtestd)))
```

## 4.2 Stabdiagramm

Steckbrief siehe Seite 65
Es wird ein Stabdiagramm in zwei Varianten angeboten. Zum einem mit maximaler Nutzung in y-Richtung (0 – max(y)), zum andern mit Normierung auf 1. Letzteres ist die Defaulteinstellung.

### Labor Information

43 ⟨*Stabdiagramm.Labor* 43⟩≡

```
##FunctionName#Stabdiagramm
##FunctionDescription#Stabdiagramm einer diskreten Haeufigkeitstabelle
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als diskrete Haeufigkeitstabelle
##FunctionDescription#MYN - == F y-Intervall [0 , max(y)]
##FunctionDescription#      == T y-Intervall [0 , 1]    (Default)
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#Stabdiagramm(x), Stabdiagramm{x,T}, Stabdiagramm(x,,F), ...
```

This code is used in chunk 44.

### Struktur Stabdiagramm

44 ⟨*start* 1⟩+≡

```
Stabdiagramm <- function(x,MNY=T,SIL=T){
⟨Stabdiagramm.Labor 43⟩
⟨Stabdiagramm.Init 45⟩
if (GO) {
  ⟨Stabdiagramm.dHtab 46⟩
}else{
  ⟨Stabdiagramm.Error 47⟩
  }
invisible() }
```

**Refinements Stabdiagramm**

*Stabdigramm.Init*

45 ⟨*Stabdiagramm.Init* 45⟩≡

```
GO     <- F
xclass <- class(x)
if ( length(xclass) > 1 && xclass[1] == "dHtab")   GO <- T
```

This code is used in chunk 44.

*Stabdigramm.dHtab*

46 ⟨*Stabdiagramm.dHtab* 46⟩≡

```
xx    <- x[,1]
yy    <- x[,3]
xmima <- c(xx[1],xx[length(xx)])
if (MNY == F)  ymima <- c(0,max(yy))
if (MNY == T)  ymima <- c(0,1)
plot(xmima,ymima,type="n",xlab="xi",ylab="ni/n")
title("Stabdiagramm")
segments(xx,0,xx,yy)
```

This code is used in chunk 44.

*Stabdigramm.Error*

47 ⟨*Stabdiagramm.Error* 47⟩≡

```
if(SIL == F) cat("\nStabdiagramm: Keine diskrete Haeufigkeitstabelle!\n")
```

This code is used in chunk 44.

### 4.2.1 Test Stabdiagramm

48 ⟨* 10⟩+≡

```
cat("\nTest Stabdiagramm\n")
ytest <- DiskHaeuf(xtestd)
cat("\nStabdiagramm(ytest)\n")
Stabdiagramm(ytest)
cat("\nStabdiagramm(xtestd)\n")
Stabdiagramm(xtestd)
```

# 5 Kontinuierliche Häufigkeitsverteilung

## 5.1 Häufigkeitstabelle

Steckbrief siehe Seite 62
Erstellt eine Häufigkeitstabelle für den kontinuierlichen Fall aus einer Urliste. Andere Eingabe wird zurückgewiesen. NAs werden entfernt.
Die kontinuierliche Häufigkeitstabelle `x` hat folgende Struktur.

| | |
|---|---|
| `x[,1]` | $x^{\star}_{i-1}$ |
| `x[,2]` | $x^{\star}_{i}$ |
| `x[,3]` | $n_i$ |
| `x[,4]` | $h_i$ |

Die Erstellung der Häufigkeitstabelle kann vom Benutzer über Parameter beinflußt werden. Es werden vier Fälle unterschieden.

| Fall | Auslöser |
|---|---|
| 1 | N, B, S alle gleich `-1`, default durch `hist` |
| 2 | B gleich gewünschter Klassengrenzen, N, S beliebig |
| 3 | N gleich gewünschter Klassenanzahl, B gleich `-1`, S beliebig |
| 4 | S gleich gewünschte Klassenbreite, B, N gleich `-1` |

### Labor Information

49 ⟨*KontHaeuf.Labor* 49⟩≡

```
##FunctionName#KontHaeuf
##FunctionDescription#Berechnet kontinuierliche Haeufigkeitstabelle
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor, Urliste oder Rangwertreihe
##FunctionDescription#   NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#N: Anzahl Klassen
##FunctionDescription#   Default: provided by hist
##FunctionDescription#B: Vektor der Klassengrenzen
##FunctionDescription#   Default: provided by hist
##FunctionDescription#S: Klassenbreite
##FunctionDescription#   Default: provided by hist
##FunctionDescription#SIL: - == F arbeitet mit Ansage
##FunctionDescription#       == T arbeitet ohne Ansage   (Default)
##FunctionDescription#sf:  - == Anzahl signifikanter Stellen (Default 2)
##FunctionDescription# last change 28.01.03
##FunctionSyntax#KontHaeuf(x), KontHaeuf(x,N), KontHaeuf(x,,B), ...
```

This code is used in chunk 50.

### Struktur KontHaeuf

50 ⟨*start* 1⟩+≡

```
KontHaeuf <- function(x, N=-1, B=-1, S=-1, SIL=T, sf=2) {
⟨KontHaeuf.Labor 49⟩
⟨KontHaeuf.Init 51⟩
if (GO) {
  ⟨KontHaeuf.Urliste 52⟩
}else{
  ⟨KontHaeuf.Error 59⟩
  }
return(erg)
invisible() }
```

### Refinements KontHaeuf

*KontHaeuf.Init*

51 ⟨*KontHaeuf.Init* 51⟩≡

```
erg <- NA
GO <- F
if (is.vector(x) && is.numeric(x)) GO <- T
```

This code is used in chunk 50.

*KontHaeuf.Urliste*

52 ⟨*KontHaeuf.Urliste* 52⟩≡

```
⟨KontHaeuf.Urliste.NA 53⟩
⟨KontHaeuf.Urliste.Fallbestimmung 54⟩
⟨KontHaeuf.Urliste.Fallabhandlung 55⟩
⟨KontHaeuf.Urliste.Ergebnisaufbereitung 56⟩
```

This code is used in chunk 50.

*KontHaeuf.Urliste.NA*

Es werden alle `NA`s aus der Urliste entfernt.

53 ⟨*KontHaeuf.Urliste.NA* 53⟩≡

```
xh    <- x[!is.na(x)]
h     <- length(x) -length(xh)
if (SIL == F && h > 0){
cat(paste("\nKontHaeuf: Es wurden",h,"NAs entfernt\n"))}
```

This code is used in chunk 52.

*KontHaeuf.Urliste.Fallbestimmung*
Priorität: B vor N oder S, N vor S

54 ⟨*KontHaeuf.Urliste.Fallbestimmung* 54⟩≡

```
sw <- 1
if(length(B) > 1)                                  sw <- 2
if((N != -1)  && (length(B) == 1))                 sw <- 3
if((N == -1)  && (length(B) == 1) && (S != -1)) sw <- 4
```

This code is used in chunk 52.

*KontHaeuf.Urliste.Fallabhandlung*

55 ⟨*KontHaeuf.Urliste.Fallabhandlung* 55⟩≡

```
if(sw == 1){ h <- hist(xh, plot = F) }
if(sw == 2){ h <- hist(xh, breaks = B, plot = F) }
if(sw == 3){
            ⟨KontHaeuf.Fall3 57⟩
           }
if(sw == 4){
            ⟨KontHaeuf.Fall4 58⟩
           }
```

This code is used in chunk 52.

*KontHaeuf.Urliste.Ergebnisaufbereitung*

56 ⟨*KontHaeuf.Urliste.Ergebnisaufbereitung* 56⟩≡

```
k   <- length(h$breaks)
n   <- sum(h$counts)
bsm <- zapsmall(h$breaks)
hsm <- zapsmall(h$counts/n)
erg <- cbind(bsm[1:(k - 1)], bsm[2:k], h$counts, hsm)
null <- NULL
dimnames(erg) <- list(null, c("xsi-1", "xsi", "ni", "ni/n"))
erg <- kHtab(erg)
```

This code is used in chunk 52.

*KontHaeuf.Fall3*
Die Extremwert werden auf- bzw. abgerundet (Anzahl signifikanter Stellen über `sf` fetgelegt), damit es *schönere* Klassengrenzen gibt. Die Funktionen `xo.nice` bzw. `xu.nice` erledigen die Arbeit.

57 ⟨*KontHaeuf.Fall3* 57⟩≡
```
xh <- sort(xh)
xu <- xu.nice(xh[1],sf)
xo <- xo.nice(xh[length(xh)],sf)
ra <- xo - xu
xd <- ra/N
xb <- xu + (0:N)*xd
h  <- hist(xh,breaks = xb, plot = F)
```
This code is used in chunk 55.

*KontHaeuf.Fall4*
Die Extremwert werden auf- bzw. abgerundet (Anzahl signifikanter Stellen über `sf` fetgelegt), damit es *schönere* Klassengrenzen gibt. Die Funktionen `xo.nice` bzw. `xu.nice` erledigen die Arbeit. Dann wird aus der Spannweite `ra` die Anzahl `nz` Klassen der Breite `S` bestimmt, die zur Überdeckung der Spannweite benötigt werden. Der Überschuß von `nz*S` über `ra` wird symmetrisch rechts und links aufgeteilt.

58 ⟨*KontHaeuf.Fall4* 58⟩≡
```
xh <- sort(xh)
xu <- xu.nice(xh[1],sf)
xo <- xo.nice(xh[length(xh)],sf)
ra <- xo - xu
nz <- ceiling(ra/S)
dd <- nz*S-ra
xu <- xu-dd/2
xb <- xu+(0:nz)*S
h  <- hist(xh,breaks = xb, plot = F)
```
This code is used in chunk 55.

*KontHaeuf.Error*

59 ⟨*KontHaeuf.Error* 59⟩≡
```
if(SIL == F) cat("\nKontHaeuf: Kein zulaessiger Datensatz!\n")
```
This code is used in chunk 50.

### 5.1.1 xo.nice

Wird von `KonHaeuf` benötigt. Erzeugt schöne Form einer oberen Grenze.

### Labor Information

60 ⟨*xo.nice.Labor* 60⟩≡

```
##FunctionName#xo.nice
##FunctionDescription#wird in KontHaeuf benoetigt
##FunctionDescription#Parameter:~
##FunctionDescription#xo: Obergrenze
##FunctionDescription#sf: signifikante Stellen  (Default 2)
##FunctionDescription#Value:~
##FunctionDescription#Schoene Form von xo
##FunctionDescription# last change 28.01.03
##FunctionSyntax#xo.nice(xo), xo.nice(xo,sf=3), ...
```

This code is used in chunk 61.

### Struktur xo.nice

61 ⟨*start* 1⟩+≡

```
xo.nice <- function(xo,sf=2) {
⟨xo.nice.Labor 60⟩
xo.a <- abs(xo)
xo.s <- sign(xo)
xo.e <- floor(log10(xo.a))
xo.z <- xo.a*10^(sf-1-xo.e)
if(xo.s==1)  xo.z <- ceiling(xo.z)
if(xo.s==-1) xo.z <- floor(xo.z)
xo.z <-xo.s*xo.z*10^(xo.e-sf+1)
return(xo.z) }
SAVE.r("xo.nice")
```

### 5.1.2 xu.nice

Wird von `KonHaeuf` benötigt. Erzeugt schöne Form einer unteren Grenze.

#### Labor Information

62 ⟨*xu.nice.Labor* 62⟩≡

```
##FunctionName#xu.nice
##FunctionDescription#wird in KontHaeuf benoetigt
##FunctionDescription#Parameter:~
##FunctionDescription#xu: Untergrenze
##FunctionDescription#sf: signifikante Stellen  (Default 2)
##FunctionDescription#Value:~
##FunctionDescription#Schoene form von xu
##FunctionDescription# last change 28.01.03
##FunctionSyntax#xu.nice(xu), xu.nice(xu,sf=3), ...
```

This code is used in chunk 63.

#### Struktur xu.nice

63 ⟨*start* 1⟩+≡

```
xu.nice <- function(xu,sf=2) {
⟨xu.nice.Labor 62⟩
xu.a <- abs(xu)
xu.s <- sign(xu)
xu.e <- floor(log10(xu.a))
xu.z <- xu.a*10^(sf-1-xu.e)
if(xu.s==1)  xu.z <- floor(xu.z)
if(xu.s==-1) xu.z <- ceiling(xu.z)
xu.z <-xu.s*xu.z*10^(xu.e-sf+1)
return(xu.z) }
SAVE.r("xu.nice")
```

### 5.1.3 Test KontHaeuf

64 ⟨* 10⟩+≡
```
cat("\nErzeugung kontinuierlicher Haeufigkeitstabellen 1\n")
cat("\nKontHaeuf(xtestk,N=5)\n")
print(KontHaeuf(xtestk,N=5))
cat("\nKontHaeuf(xtestk)\n")
print(KontHaeuf(xtestk))
cat("\nKontHaeuf(xtestk,S=0.25))\n")
print(KontHaeuf(xtestk,S=0.25))
```

65 ⟨* 10⟩+≡
```
cat("\nErzeugung kontinuierlicher Haeufigkeitstabellen 2\n")
cat("\nKontHaeuf(xtestb)\n")
print(KontHaeuf(xtestb))
cat("\nKontHaeuf(xtestk)\n")
print(KontHaeuf(xtestk))
cat("\nKontHaeuf(c(NA,NA,xtestk))\n")
print(KontHaeuf(c(NA,NA,xtestk)))
```

## 5.2 Histogramm

Steckbrief siehe Seite 62
Es wird ein Histogramm angeboten.

### Labor Information

66 ⟨*Histogramm.Labor* 66⟩≡
```
##FunctionName#Histogramm
##FunctionDescription#Histogramm einer kontinuierlichen Haeufigkeitstabelle
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als kontinuierliche Haeufigkeitstabelle
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#Histogramm(x), Histogramm(x,F)
```
This code is used in chunk 67.

### Struktur Histogramm

67 ⟨*start* 1⟩+≡

```
Histogramm <- function(x,SIL=T){
⟨Histogramm.Labor 66⟩
⟨Histogramm.Init 68⟩
if (GO) {
 ⟨Histogramm.kHtab 69⟩
}else{
  ⟨Histogramm.Error 70⟩
  }
invisible() }
```

### Refinements Histogramm

*Histogramm.Init*

68 ⟨*Histogramm.Init* 68⟩≡

```
   GO     <- F
   xclass <- class(x)
   if ( length(xclass) > 1 && xclass[1] == "kHtab")   GO <- T
```

This code is used in chunk 67.

*Histogramm.kHtab*

69 ⟨*Histogramm.kHtab* 69⟩≡

```
   x1    <- x[,1]
   x2    <- x[,2]
   xd    <- x2-x1
   nn    <- length(x1)
   xx    <- c(x1,x2[nn])
   yy    <- x[,4]
   yy    <- yy/xd
   xmima <- c(x1[1],x2[nn])
   ymima <- c(0,max(yy))
   plot(xmima,ymima,type="n",xlab="xi",ylab="f-dach")
   title("Histogramm")
   segments(x1,0,x1,yy)
   segments(x2,0,x2,yy)
   segments(x1,yy,x2,yy)
   lines(c(x1[1],x2[nn]),c(0,0))
```

This code is used in chunk 67.

*Histogramm.Error*

70 ⟨*Histogramm.Error* 70⟩≡

```
    if(SIL==F) cat("\nHistogramm: Keine kontinuierliche Haeufigkeitstabelle!\n")
```

This code is used in chunk 67.

### 5.2.1 Test Histogramm

71 ⟨* 10⟩+≡

```
cat("\nTest Histogramm\n")
ytest <- KontHaeuf(xtestk)
cat("\nHistogramm(ytest)\n")
Histogramm(ytest)
cat("\nHistogramm(xtestd)\n")
Histogramm(xtestd)
```

# 6 Empirische Verteilungsfunktion

## 6.1 Graph: EmpVert

Steckbrief siehe Seite 61
Die Funktion `Emp.vert` erstellt einen Graphen einer empirischen Verteilungsfunktion. Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.

### Labor Information

72 ⟨*EmpVert.Labor* 72⟩≡

```
##FunctionName#EmpVert
##FunctionDescription#Graph der empirischen Verteilungsfunktion
##FunctionDescription#Parameter:~
##FunctionDescription#x: Daten als Vektor, Urliste oder Rangwertreihe
##FunctionDescription#   NA‘s zugelassen, werden vor Zeichnung entfernt
##FunctionDescription#MerkName: Bezeichnung des Merkmals (string), Default "x"
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription# last change 27.01.03
##FunctionSyntax#EmpVert(x), EmpVert(x,MerkName), EmpVert(x,,F), ...
```

This code is used in chunk 73.

### Struktur EmpVert

73 ⟨*start* 1⟩+≡

```
EmpVert <- function(x,MerkName="x",SIL=T) {
⟨EmpVert.Labor 72⟩
⟨EmpVert.Init 74⟩
if (GO) {
  if (is.vector(x)) {
    ⟨EmpVert.Urliste 75⟩
  }else{
    if (xclass[1] == "dHtab") {
      ⟨EmpVert.dHtab 76⟩
      }
    if (xclass[1] == "kHtab") {
      ⟨EmpVert.kHtab 77⟩
      }  }
  ⟨EmpVert.Plot 78⟩
}else{
  ⟨EmpVert.Error 79⟩
  }
invisible() }
```

### Refinements EmpVert

*EmpVert.Init*

74 ⟨*EmpVert.Init* 74⟩≡

```
GO <- F
xclass <- class(x)
if (is.vector(x)       && is.numeric(x))         GO <- T
if (length(xclass) > 1 && xclass[2] == "Htab")   GO <- T
```

This code is used in chunk 73.

*EmpVert.Urliste*

75 ⟨*EmpVert.Urliste* 75⟩≡

```
SPRUNG <- T
xh    <- x[!is.na(x)]
n     <- length(xh)
h     <- length(x)-n
if(SIL == F && h > 0){
cat(paste("\nEmpVert: Es wurden", h, "NAs entfernt\n"))}
xh    <- sort(xh)
# entferne Duplikate
x1    <- c(xh[-1],2*xh[n])
ll    <- xh!=x1
xp    <- xh[ll]
# ermittle Sprunghoehe
yp    <- (1:n)[ll]
yp    <- yp/n
# Vorbereitung plot
nn    <- length(xp)
spann <- (xp[nn]-xp[1])*0.1
xmima <- c(xp[1]-spann,xp[nn]+spann)
ymima <- c(0,1)
```

This code is used in chunk 73.

*EmpVert.dHtab*

76 ⟨*EmpVert.dHtab* 76⟩≡

```
SPRUNG <- T
yp     <- cumsum(x[,3])
xp     <- x[,1]
nn     <- length(xp)
spann  <- (xp[nn]-xp[1])*0.1
xmima  <- c(xp[1]-spann,xp[nn]+spann)
ymima  <- c(0,1)
```

This code is used in chunk 73.

*EmpVert.kHtab*

77 ⟨*EmpVert.kHtab* 77⟩≡

```
SPRUNG <- F
yp     <- cumsum(x[,4])
x1     <- x[,1]
x2     <- x[,2]
nn     <- length(x1)
spann  <- (x2[nn]-x1[1])*0.1
xmima  <- c(x1[1]-spann,x2[nn]+spann)
ymima  <- c(0,1)
```

This code is used in chunk 73.

*EmpVert.Plot*

78 ⟨*EmpVert.Plot* 78⟩≡

```
# eigentlicher plot
plot(xmima,ymima,type="n",xlab=MerkName,ylab="F-dach")
title("Empirische Verteilungsfunktion")
if (SPRUNG)  segments(xp,yp,c(xp[-1],xp[nn]+spann),yp)
if (!SPRUNG) lines(c(x1,x2[nn],x2[nn]+spann),c(0,yp,yp[nn]))
```

This code is used in chunk 73.

*EmpVert.Error*

79 ⟨*EmpVert.Error* 79⟩≡

```
if(SIL == F) cat("\nEmpVert: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 73.

### 6.1.1 Test EmpVert

80 ⟨* 10⟩+≡

```
cat("\nEmpirische Verteilung: Urliste\n")
EmpVert(xtestb)
EmpVert(xtestd)
```

81 ⟨* 10⟩+≡

```
cat("\nEmpirische Verteilung: DiskHaeuf\n")
EmpVert(DiskHaeuf(xtestd))
```

82 ⟨* 10⟩+≡

```
cat("\nEmpirische Verteilung: KontHaeuf\n")
EmpVert(KontHaeuf(xtestk))
```

## 6.2 Wert: pemp

Steckbrief siehe Seite 59

Die Funktion `pemp` berechnet den Wert einer empirischen Verteilungsfunktion an der Stelle $x$. Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.
Notation in Analoge zu dname, pname, qname, rname in S und R.

83 ⟨*pemp.Labor* 83⟩≡

```
##FunctionName#pemp
##FunctionDescription#Parameter:~
##FunctionDescription#xq: Stelle
##FunctionDescription#x:  - Daten als Urliste,
##FunctionDescription#       NAs werden vor Berechnung entfernt
##FunctionDescription#      Daten alsdHtab, kHtab
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#empirische Verteilungsfunktion  an der Stelle xq
##FunctionDescription# last change 27.01.03
##FunctionSyntax#pemp(xq,x), pemp(xq,x,F)
```

This code is used in chunk 84.

### Struktur pemp

84 ⟨*start* 1⟩+≡

```
pemp <- function(xq,x,SIL=T) {
⟨pemp.Labor 83⟩
⟨pemp.Init 85⟩
if (GO) {
  if (is.vector(x)) {
    ⟨pemp.Urliste 86⟩
  }else{
    if (xclass[1] == "dHtab") {
      ⟨pemp.dHtab 87⟩
      }
    if (xclass[1] == "kHtab") {
      ⟨pemp.kHtab 88⟩
      } }
}else{
  ⟨pemp.Error 89⟩
  }
return(erg)
invisible() }
```

### Refinements pemp

*pemp.Init*

85 ⟨*pemp.Init* 85⟩≡

```
erg    <- NA
GO     <- F
xclass <- class(x)
if (is.vector(x)        && is.numeric(x))         GO <- T
if ( length(xclass) > 1 && xclass[2] == "Htab")   GO <- T
```

This code is used in chunk 84.

*pemp.Urliste*

86 ⟨*pemp.Urliste* 86⟩≡

```
xh  <- x[!is.na(x)]
nn  <- length(xh)
h   <- length(x) - nn
if(SIL == F && h > 0){
cat(paste("\npemp: Es wurden", h, "NAs entfernt\n"))}
erg <- sum(rep(1,nn)[xh <= xq])/nn
```

This code is used in chunk 84.

*pemp.dHtab*

87 ⟨*pemp.dHtab* 87⟩≡

```
xx <- x[,1]
nn <- length(xx)
yy <- x[,3]
lo <- T
if(xq<x[1]) { erg <- 0
              lo <- F }
if(xq>x[nn]) { erg <- 1
               lo <- F }
if(lo) {
  xl  <- xx <= xq
  erg <- sum(yy[xl]) }
```

This code is used in chunk 84.

*pemp.kHtab*

88 ⟨*pemp.kHtab* 88⟩≡

```
di <- x[,2]-x[,1]
hi <- x[,4]
nn <- length(hi)
lo <- T
if(xq<=x[1,1]) { erg <- 0
                 lo <- F }
if(xq>x[nn,2]) { erg <- 1
                 lo <- F }
if(lo) {
  h <- 0
  i <- 1
  while ( xq >= x[i,1]) {
    h <- h+ hi[i]
    i <- i+1 }
  erg <- as.vector(h +(xq-x[i,1])*hi[i]/di[i]) }
```

This code is used in chunk 84.

*pemp.Error*

89 ⟨*pemp.Error* 89⟩≡

```
if(SIL == F) cat("\npemp: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 84.

### 6.2.1 Test pemp

90 ⟨* 10⟩+≡

```
cat("\nTest pemp\n")
cat("\n pemp(4,1:10)\n")
print(pemp(4,1:10))
cat("\n pemp(4,c(NA,1:10))\n")
print(pemp(4,c(NA,1:10)))
print(pemp(0,KontHaeuf(rnorm(100))))
print(pemp(0,DiskHaeuf(rpois(100,2))))
```

## 6.3 Prozentpunkt: qemp

Steckbrief siehe Seite 60

Die Funktion `qemp` berechnet den Prozentpunkt zum Anteil $p$ für eine empirische Verteilungsfunktion. Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.
Im Medianmodus stimmt das Ergebnis bei einer Urliste und $p = 0.5$ mit dem Ergebnis von `Median` überein.
Notation in Analoge zu dname, pname, qname, rname in S und R.

91 ⟨*qemp.Labor* 91⟩≡

```
##FunctionName#qemp
##FunctionDescription#Parameter:~
##FunctionDescription#p: Anteil
##FunctionDescription#x: - Daten als Urliste,
##FunctionDescription#      NAs werden vor Berechnung entfernt
##FunctionDescription#     Daten als dHtab oder kHtab
##FunctionDescription#MED - == F arbeitet nach Formel (Default)
##FunctionDescription#      == T arbeitet wie Median bei p=0.5 und Urliste
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#Prozentpunkt f\"ur p
##FunctionDescription# last change 13.02.03
##FunctionSyntax#qemp(p,x), qemp(p,x,F), qemp(0.5,x,T,T)
```

This code is used in chunk 92.

### Struktur qemp

92 ⟨*start* 1⟩+≡

```
qemp <- function(p,x,MED=F,SIL=T) {
⟨qemp.Labor 91⟩
⟨qemp.Init 93⟩
if (GO) {
  if (is.vector(x)) {
    ⟨qemp.Urliste 94⟩
  }else{
    if (xclass[1] == "dHtab") {
      ⟨qemp.dHtab 95⟩
      }
    if (xclass[1] == "kHtab") {
      ⟨qemp.kHtab 96⟩
      } }
}else{
  ⟨qemp.Error 97⟩
  }
return(erg)
invisible() }
```

### Refinements qemp *qemp.Init*

93 ⟨*qemp.Init* 93⟩≡

```
   erg    <- NA
   GO     <- F
   xclass <- class(x)
   if (is.vector(x)        && is.numeric(x))         GO <- T
   if ( length(xclass) > 1 && xclass[2] == "Htab")   GO <- T
   if ( p <= 0 || p >= 1)  { GO <- F
                             if(SIL == F) cat("\nqemp: p unzulaessig\n")}
```

This code is used in chunk 92.

*qemp.Urliste*

94 ⟨*qemp.Urliste* 94⟩≡

```
xh  <- x[!is.na(x)]
nn  <- length(xh)
h   <- length(x) - nn
if(SIL == F && h > 0){
  cat(paste("\nqemp: Es wurden", h, "NAs entfernt\n"))}
xh  <- sort(xh)
if((zapsmall(p-0.5)== 0) && (MED==T)) {
if(SIL==F) cat("\nqemp: Median-Modus an\n")
np  <- nn*p
n1  <- ceiling(np)
h   <- 0
if(nn==(2*n1)) h <- 1
n2  <- n1+h
erg <- (xh[n1]+xh[n2])/2

}else{
 Fi  <- (1:nn)/nn
 i   <- 1
 while (p > Fi[i] ) i <- i+1
 erg <- as.vector(xh[i]) }
```

This code is used in chunk 92.

*qemp.dHtab*

95 ⟨*qemp.dHtab* 95⟩≡

```
hi  <- x[,3]
Fi  <- cumsum(hi)
i   <- 1
while (p > Fi[i] ) i <- i+1
erg <- as.vector(x[i,1])
```

This code is used in chunk 92.

*qemp.kHtab*

96 ⟨*qemp.kHtab* 96⟩≡

```
di  <- x[,2]-x[,1]
hi  <- x[,4]
xi  <- x[,1]
Fi  <- c(0,cumsum(hi))

i   <- 1
while (p > Fi[i+1] ) i <- i+1
erg <- as.vector(xi[i]  + (p - Fi[i])/hi[i]*di[i])
```

This code is used in chunk 92.

*qemp.Error*

97 ⟨*qemp.Error* 97⟩≡

```
if(SIL == F) cat("\nqemp: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 92.

### 6.3.1 Test qemp

98 ⟨* 10⟩+≡

```
cat("\nTest qemp\n")
cat("\nqemp(0.1,xtestb\n")
print(qemp(0.1,xtestb))
cat("\nqemp(0.4,1:10)\n")
print(qemp(0.4,1:10))
cat("\nqemp(1.2,1:10)\n")
print(qemp(1.2,1:10))
cat("\n qemp(0.4,c(NA,1:10))\n")
print(qemp(0.4,c(NA,1:10)))
cat("\nqemp(0.6,DiskHaeuf(rpois(100,2)))\n")
ttt <- DiskHaeuf(rpois(100,2))
print(ttt)
print(qemp(0.6,ttt))
cat("\nqemp(0.5,KontHaeuf(rnorm(100)))\n")
ttt <- KontHaeuf(rnorm(100))
print(ttt)
print(qemp(0.5,ttt))
cat("\nqemp(0.5,1:10)\n")
print(qemp(0.5,1:10))
cat("\nqemp(0.5,1:10,T)\n")
print(qemp(0.5,1:10,T))
```

# 7 Maßzahlen

## 7.1 Median

Steckbrief siehe Seite 63

Die Funktion `Median` arbeitet abgesehen von der Urliste wie `qemp` für den Fall $p = 0.5$. Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.

99 ⟨*Median.Labor* 99⟩≡

```
##FunctionName#Median
##FunctionDescription#Parameter:~
##FunctionDescription#x: - Daten als Urliste,
##FunctionDescription#      NAs werden vor Berechnung entfernt
##FunctionDescription#     Daten als dHtab oder kHtab
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#Median
##FunctionDescription# last change 13.02.03
##FunctionSyntax#Median(x), Median(x,F)
```

This code is used in chunk 100.

### Struktur Median

100 ⟨*start* 1⟩+≡

```
Median <- function(x,SIL=T) {
⟨Median.Labor 99⟩
⟨Median.Init 101⟩
if (GO) {
  if (is.vector(x)) {
    ⟨Median.Urliste 102⟩
  }else{
    if (xclass[1] == "dHtab") {
      ⟨Median.dHtab 103⟩
      }
    if (xclass[1] == "kHtab") {
      ⟨Median.kHtab 104⟩
      } }
}else{
  ⟨Median.Error 105⟩
  }
return(erg)
invisible() }
```

### Refinements Median *Median.Init*

101 ⟨*Median.Init* 101⟩≡

```
   p      <- 0.5
   erg    <- NA
   GO     <- F
   xclass <- class(x)
   if (is.vector(x)        && is.numeric(x))         GO <- T
   if ( length(xclass) > 1 && xclass[2] == "Htab")   GO <- T
```

This code is used in chunk 100.

*Median.Urliste*

102 ⟨*Median.Urliste* 102⟩≡

```
xh  <- x[!is.na(x)]
nn  <- length(xh)
h   <- length(x) - nn
if(SIL == F && h > 0){
  cat(paste("\nMedian: Es wurden", h, "NAs entfernt\n"))}
xh  <- sort(xh)
np  <- nn*p
n1  <- ceiling(np)
h   <- 0
if(nn==(2*n1)) h <- 1
n2  <- n1+h
erg <- (xh[n1]+xh[n2])/2
```

This code is used in chunk 100.

*Median.dHtab*

103 ⟨*Median.dHtab* 103⟩≡

```
hi  <- x[,3]
Fi  <- cumsum(hi)
i   <- 1
while (p > Fi[i] ) i <- i+1
erg <- as.vector(x[i,1])
```

This code is used in chunk 100.

*Median.kHtab*

104 ⟨*Median.kHtab* 104⟩≡

```
di  <- x[,2]-x[,1]
hi  <- x[,4]
xi  <- x[,1]
Fi  <- c(0,cumsum(hi))

i   <- 1
while (p > Fi[i+1] ) i <- i+1
erg <- as.vector(xi[i]  + (p - Fi[i])/hi[i]*di[i])
```

This code is used in chunk 100.

*Median.Error*

105 ⟨*Median.Error* 105⟩≡

```
if(SIL == F) cat("\nMedian: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 100.

### 7.1.1 Test Median

106 ⟨* 10⟩+≡

```
cat("\nTest Median\n")
cat("\nMedian(1:5)\n")
print(Median(1:5))
cat("\nMedian(1:10)\n")
print(Median(1:10))
cat("\n Median(c(NA,1:10))\n")
print(Median(c(NA,1:10)))
cat("\nMedian(DiskHaeuf(rpois(100,2)))\n")
ttt <- DiskHaeuf(rpois(100,2))
print(ttt)
print(Median(ttt))
cat("\nMedian(KontHaeuf(rnorm(100)))\n")
ttt <- KontHaeuf(rnorm(100))
print(ttt)
print(Median(ttt))
```

## 7.2 Arithmetisches Mittel

Steckbrief siehe Seite 64
Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.

### Labor Information

107 ⟨*Mittelwert.Labor* 107⟩≡

```
##FunctionName#Mittelwert
##FunctionDescription#Arithmetisches Mittel
##FunctionDescription#Parameter:~
##FunctionDescription#x:  - Vektor: Urliste oder Rangwertreihe
##FunctionDescription#      NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#    - Daten als diskrete Haeufigkeitstabelle
##FunctionDescription#    - Daten als kontinuierliche Haeufigkeitstabelle
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage   (Default)
##FunctionDescription#Value:~
##FunctionDescription#Mittelwert von x oder NA
##FunctionDescription# last change 26.01.03
##FunctionSyntax#Mittelwert(x), Mittelwert(x,F)
```

This code is used in chunk 108.

### Struktur Mittelwert

108 ⟨*start* 1⟩+≡

```
Mittelwert <- function(x,SIL=T) {
⟨Mittelwert.Labor 107⟩
⟨Mittelwert.Init 109⟩
if (GO) {
  if (is.vector(x)) {
   ⟨Mittelwert.Urliste 110⟩
  }else{
    if (xclass[1] == "dHtab") {
      ⟨Mittelwert.dHtab 111⟩
      }
    if (xclass[1] == "kHtab") {
      ⟨Mittelwert.kHtab 112⟩
      } }
}else{
  ⟨Mittelwert.Error 113⟩
  }
return(erg)
invisible() }
```

### Refinements Mittelwert

*Mittelwert.Init*

109 ⟨*Mittelwert.Init* 109⟩≡

```
  erg    <- NA
  GO     <- F
  xclass <- class(x)
  if (is.vector(x)       && is.numeric(x))         GO <- T
  if ( length(xclass) >1 && xclass[2] == "Htab")   GO <- T
```

This code is used in chunk 108.

*Mittelwert.Urliste*

110 ⟨*Mittelwert.Urliste* 110⟩≡

```
  if(SIL == F) cat("\n Mittelwert aus Urliste\n")
  xh <- x[!is.na(x)]
  h  <- length(x) - length(xh)
  if(SIL == F && h > 0) {
    cat(paste("\nMittelwert: Es wurden", h, "NAs entfernt\n")) }
  erg <- mean(xh)
```

This code is used in chunk 108.

*Mittelwert.dHtab*

111 ⟨*Mittelwert.dHtab* 111⟩≡

```
if(SIL==F) cat("\n Mittelwert aus diskreter Haeufigkeitstabelle\n")
erg <- sum(x[, 1] * x[, 3])
```

This code is used in chunk 108.

*Mittelwert.kHtab*

112 ⟨*Mittelwert.kHtab* 112⟩≡

```
if(SIL==F){
  cat("\n Mittelwert aus kontinuierlicher Haeufigkeitstabelle\n")}
erg <- sum(((x[, 2] + x[, 1]) * 0.5) * x[, 4])
```

This code is used in chunk 108.

*Mittelwert.Error*

113 ⟨*Mittelwert.Error* 113⟩≡

```
if(SIL == F) cat("\nMittelwert: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 108.

### 7.2.1 Test Mittelwert

114 ⟨* 10⟩+≡

```
cat("\nTest Mittelwert\n")
cat("\Mittelwert(xtestb)\n")
print(Mittelwert(xtestb))
cat("\nMittelwert(matrix(1:6,2,3))\n")
print(Mittelwert(matrix(1:6,2,3)))
cat("\Mittelwert(xtestd)\n")
print(Mittelwert(xtestd))
cat("\nMittelwert(c(NA,NA,xtestk))\n")
print(Mittelwert(c(NA,NA,xtestk)))
cat("\nMittelwert(DiskHaeuf(xtestd))\n")
print(Mittelwert(DiskHaeuf(xtestd)))
cat("\nMittelwert(DiskHaeuf(xtestk))\n")
print(Mittelwert(KontHaeuf(xtestk)))
```

## 7.3 Varianz

Steckbrief siehe Seite 66
Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.

### Labor Information

115 ⟨*Varianz.Labor* 115⟩≡

```
##FunctionName#Varianz
##FunctionDescription#Varianz
##FunctionDescription#Parameter:~
##FunctionDescription#x:  - Vektor: Urliste oder Rangwertreihe
##FunctionDescription#      NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#    - Daten als diskrete Haeufigkeitstabelle
##FunctionDescription#    - Daten als kontinuierliche Haeufigkeitstabelle
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage  (Default)
##FunctionDescription#Value:~
##FunctionDescription#Varianz von x oder NA
##FunctionDescription# last change 04.02.03
##FunctionSyntax#Varianz(x), Varianz(x,F)
```

This code is used in chunk 116.

### Struktur Varianz

116 ⟨*start* 1⟩+≡

```
Varianz<- function(x,SIL=T) {
⟨Varianz.Labor 115⟩
⟨Varianz.Init 117⟩
if (GO) {
  if (is.vector(x)) {
    ⟨Varianz.Urliste 118⟩
  }else{
    if (xclass[[1]] == "dHtab") {
      ⟨Varianz.dHtab 119⟩
      }
    if (xclass[[1]] == "kHtab") {
      ⟨Varianz.kHtab 120⟩
      } }
}else{
  ⟨Varianz.Error 121⟩
  }
return(erg)
invisible() }
```

**Refinements Varianz**

*Varianz.Init*

117 ⟨*Varianz.Init* 117⟩≡

```
erg    <- NA
GO     <- F
xclass <- class(x)
if (is.vector(x)        && is.numeric(x))           GO <- T
if ( length(xclass) > 1 && xclass[[2]] == "Htab")   GO <- T
```

This code is used in chunk 116.

*Varianz.Urliste*

118 ⟨*Varianz.Urliste* 118⟩≡

```
if(SIL == F) cat("\n Varianz aus Urliste\n")
xh <- x[!is.na(x)]
h <- length(x) - length(xh)
if(SIL == F && h > 0){
  cat(paste("\nVarianz: Es wurden", h, "NAs entfernt\n"))}
erg <- var(xh)
```

This code is used in chunk 116.

*Varianz.dHtab*

119 ⟨*Varianz.dHtab* 119⟩≡

```
if(SIL == F){
  cat("\n Varianz aus diskreter Haeufigkeitstabelle\n")}
n   <- sum(x[,2])
xq  <- sum(x[, 1] * x[, 2])/n
erg <- (sum( ((x[,1]-xq )^2 ) * x[,2] ))/(n-1)
```

This code is used in chunk 116.

*Varianz.kHtab*

120 ⟨*Varianz.kHtab* 120⟩≡

```
if(SIL == F){
  cat("\n Varianz aus kontinuierlicher Haeufigkeitstabelle\n")}
n   <- sum(x[,3])
xmi <- (x[, 2] + x[, 1]) * 0.5
xq  <- sum(xmi* x[, 3])/n
erg <- (sum( ((xmi-xq )^2 ) * x[,3] ) )/(n-1)
```

This code is used in chunk 116.

*Varianz.Error*

121 ⟨*Varianz.Error* 121⟩≡

```
if(SIL == F) cat("\nVarianz: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 116.

### 7.3.1 Test Varianz

122 ⟨* 10⟩+≡

```
cat("\nTest Varianz\n")
cat("\nVarianz(xtestb)\n")
print(Varianz(xtestb))
cat("\nVarianz(matrix(1:6,2,3))\n")
print(Varianz(matrix(1:6,2,3)))
cat("\nVarianz(xtestd)\n")
print(Varianz(xtestd))
cat("\Varianz(c(NA,NA,xtestk))\n")
print(Varianz(c(NA,NA,xtestk)))
cat("\nVarianz(DiskHaeuf(xtestd))\n")
print(Varianz(DiskHaeuf(xtestd)))
cat("\nVarianz(KontHaeuf(xtestk))\n")
print(Varianz(KontHaeuf(xtestk)))
```

## 7.4 MQA

Steckbrief siehe Seite 64
Arbeitet auf allen drei Eingaben: Urliste, diskrete Häufigkeitstabelle und kontinuierliche Häufigkeitstabelle. Andere Eingabe wird zurückgewiesen.

### Labor Information

123 ⟨*MQA.Labor* 123⟩≡

```
##FunctionName#MQA
##FunctionDescription#MQA
##FunctionDescription#Parameter:~
##FunctionDescription#x:  - Vektor: Urliste oder Rangwertreihe
##FunctionDescription#      NA's zugelassen, werden vor Berechnung entfernt
##FunctionDescription#    - Daten als diskrete Haeufigkeitstabelle
##FunctionDescription#    - Daten als kontinuierliche Haeufigkeitstabelle
##FunctionDescription#SIL - == F arbeitet mit Ansage
##FunctionDescription#      == T arbeitet ohne Ansage  (Default)
##FunctionDescription#Value:~
##FunctionDescription#MQA von x oder NA
##FunctionDescription# last change 04.02.03
##FunctionSyntax#MQA(x), MQA(x,F)
```

This code is used in chunk 124.

### Struktur MQA

124 ⟨*start* 1⟩+≡

```
MQA<- function(x,SIL=T) {
⟨MQA.Labor 123⟩
⟨MQA.Init 125⟩
if (GO) {
  if (is.vector(x)) {
    ⟨MQA.Urliste 126⟩
  }else{
    if (xclass[[1]] == "dHtab") {
      ⟨MQA.dHtab 127⟩
      }
    if (xclass[[1]] == "kHtab") {
      ⟨MQA.kHtab 128⟩
      } }
}else{
  ⟨MQA.Error 129⟩
  }
return(erg)
invisible() }
```

### Refinements MQA

*MQA.Init*

125 ⟨*MQA.Init* 125⟩≡

```
   erg    <- NA
   GO     <- F
   xclass <- class(x)
   if (is.vector(x)        && is.numeric(x))           GO <- T
   if ( length(xclass) > 1 && xclass[[2]] == "Htab")   GO <- T
```

This code is used in chunk 124.

*MQA.Urliste*

126 ⟨*MQA.Urliste* 126⟩≡

```
if(SIL == F) cat("\n MQA aus Urliste\n")
xh <- x[!is.na(x)]
h <- length(x) - length(xh)
if(SIL == F && h > 0){
  cat(paste("\nMQA: Es wurden", h, "NAs entfernt\n"))}
n   <- length(xh)
erg <- var(xh)*(n-1)/n
```

This code is used in chunk 124.

*MQA.dHtab*

127 ⟨*MQA.dHtab* 127⟩≡

```
if(SIL == F){
  cat("\n MQA aus diskreter Haeufigkeitstabelle\n")}
n   <- sum(x[,2])
xq  <- sum(x[, 1] * x[, 2])/n
erg <- (sum( ((x[,1]-xq )^2 ) * x[,2] ))/n
```

This code is used in chunk 124.

*MQA.kHtab*

128 ⟨*MQA.kHtab* 128⟩≡

```
if(SIL == F){
  cat("\n MQA aus kontinuierlicher Haeufigkeitstabelle\n")}
n   <- sum(x[,3])
xmi <- (x[, 2] + x[, 1]) * 0.5
xq  <- sum(xmi* x[, 3])/n
erg <- (sum( ((xmi-xq )^2 ) * x[,3] ) )/n
```

This code is used in chunk 124.

*MQA.Error*

129 ⟨*MQA.Error* 129⟩≡

```
if(SIL == F) cat("\nMQA: Keine zulaessige Eingabe!\n")
```

This code is used in chunk 124.

### 7.4.1 Test MQA

130 ⟨* 10⟩+≡

```
cat("\nTest MQA\n")
cat("\nMQA(xtestb)\n")
print(MQA(xtestb))
cat("\nMQA(matrix(1:6,2,3))\n")
print(MQA(matrix(1:6,2,3)))
cat("\nMQA(xtestd)\n")
print(MQA(xtestd))
cat("\MQA(c(NA,NA,xtestk))\n")
print(MQA(c(NA,NA,xtestk)))
cat("\nMQA(DiskHaeuf(xtestd))\n")
print(MQA(DiskHaeuf(xtestd)))
cat("\nMQA(KontHaeuf(xtestk))\n")
print(MQA(KontHaeuf(xtestk)))
```

# 8 Daten

131 ⟨*start* 1⟩+≡

```
xtestd <- rpois(50,2)
xtestk <- rnorm(100)
xtestb <- c("w","w","W")
```

# 9 Steckbriefe

Der Steckbrief wird nur für Funktionen der oberen Ebene gegeben. Die Steckbriefe sind alphabetisch geordnet.

| box.and.whisker | Box-und-Whisker-Plot | box.and.whisker |
|---|---|---|

```
box.and.whisker(x,kenn="")
```

ARGUMENTE

| | |
|---|---|
| x | liste von Datensätzen |
| kenn | Vektor der Kennungen der Datensätze (optional) |
| SIL | Arbeitmodus: = T lautlos, = F mit Information |

RESULT

Graphik

CODE

siehe Seite 9

| five.values | Fünf-Zahlen-Darstellung | five.values |
|---|---|---|

```
five.values(x,tukey=F)
```

ARGUMENTE

| | |
|---|---|
| x | Vektor: Urliste oder Rangwertreihe<br>NA's zugelassen, werden vor Berechnung entfernt |
| tukey | =F : einfache Version<br>=T : Tukey-Stil |
| SIL | Arbeitmodus: = T lautlos, = F mit Information |

RESULT

Textgraphik

CODE

siehe Seite 5

| pemp | Wert der Empirischen Verteilungsfunktion | pemp |
|---|---|---|

`pemp(xq,x)`

ARGUMENTE

| | |
|---|---|
| `xq` | Stelle |
| `x` | Daten als Vektor: Urliste oder Rangwertreihe |
| | NA's zugelassen, werden vor Berechnung entfernt |
| | Daten als diskrete Haeufigkeitstabelle |
| | Daten als kontinuierliche Haeufigkeitstabelle |
| `SIL` | Arbeitmodus: = T lautlos, = F mit Information |

RESULT

$\hat{F}(xq)$
NA bei unzulässigem Argument x

CODE

siehe Seite 39

| `qemp` | Prozentpunkt der Empirischen Verteilungsfunktion | `qemp` |
|---|---|---|

`pemp(p,x)`

ARGUMENTE

| | |
|---|---|
| `p` | Anteil |
| `x` | Daten als Vektor: Urliste oder Rangwertreihe |
| | NA's zugelassen, werden vor Berechnung entfernt |
| | Daten als diskrete Haeufigkeitstabelle |
| | Daten als kontinuierliche Haeufigkeitstabelle |
| `MED` | Medianmodus: = T an, = F aus (Default) |
| `SIL` | Arbeitmodus: = T lautlos (Default), = F mit Information |

RESULT

$\hat{F}^{-1}(p)$
NA bei unzulässigem Argument x

CODE

siehe Seite 42

| `upper.versus.lower` | Graph Obere gegen Untere Datenhälfte | `upper.versus.lower` |
|---|---|---|

`upper.versus.lower(x`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwerteihe
NA‘s zugelassen, werden vor Zeichnung entfernt

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Graphik

CODE

siehe Seite 14

| `DiskHaeuf` | Diskrete Häufigkeitsverteilung | `DiskHaeuf` |
|---|---|---|

`DiskHaeuf(x)`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwertreihe
NA‘s zugelassen, werden vor Berechnung entfernt

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Objekt der Klasse “Diskrete Häufigkeitsverteilung
vom Typ liste, dabei ist:

- Element 1 == Vektor der Merkmalsausprägungen
- Element 2 == Vektor der absoluten Häufigkeiten
- Element 3 == Vektor der relativen Häufigkeiten

NA bei unzulässigem Argument

CODE

siehe Seite 21

| `EmpVert` | Graph der Empirischen Verteilungsfunktion | `EmpVert` |
|---|---|---|

`EmpVert(x)`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwertreihe
NA's zugelassen, werden vor Berechnung entfernt
Daten als diskrete Haeufigkeitstabelle
Daten als kontinuierliche Haeufigkeitstabelle

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Graphik

CODE

siehe Seite 35

| `Histogramm` | Histogramm | `Histogramm` |
|---|---|---|

`Stabdiagramm(x)`

ARGUMENTE

`x` Daten als Objekt "Kontinuierliche Häufigkeitsverteilung

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Graphik

CODE

siehe Seite 32

| `KontHaeuf` | Kontinuierliche Häufigkeitsverteilung | `KontHaeuf` |
|---|---|---|

`KontHaeuf(x,N=-1,B=-1,S=-1)`

ARGUMENTE

| | |
|---|---|
| `x` | Daten als Vektor: Urliste oder Rangwertreihe<br>NA's zugelassen, werden vor Berechnung entfernt |
| `N` | Anzahl der Klassen |
| `B` | Vektor der Klassengrenzen |
| `S` | Klassenbreite |
| `SIL` | Arbeitmodus: = T lautlos, = F mit Information |

BEMERKUNG

Die Defaultsetzungen überlassen der R-Funktion `hist` die Bestimmung der Klassenanzahl, der Klassengrenzen bzw. der Klassenbreite.

RESULT

Objekt der Klasse "Kontinuierliche Häufigkeitsverteilung vom Typ liste, dabei ist:

- Element 1 == Vektor der Klassenuntergrenzen
- Element 2 == Vektor der Klassenobergrenzen
- Element 3 == Vektor der absoluten Häufigkeiten
- Element 4 == Vektor der relativen Häufigkeiten

NA bei unzulässigem Argument

CODE

siehe Seite 26

`Median(x)`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwertreihe
NA's zugelassen, werden vor Berechnung entfernt
Daten als diskrete Haeufigkeitstabelle
Daten als kontinuierliche Haeufigkeitstabelle

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Median von x (gemäß entsprechender Formel)
NA bei unzulässigem Argument

CODE

siehe Seite 47

| `Mittelwert` | Mittelwert | `Mittelwert` |
|---|---|---|

`Mittelwert(x)`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwertreihe
NA's zugelassen, werden vor Berechnung entfernt
Daten als diskrete Haeufigkeitstabelle
Daten als kontinuierliche Haeufigkeitstabelle

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Mittelwert von x (gemäß entsprechender Formel)
NA bei unzulässigem Argument

CODE

siehe Seite 51

| `MQA` | MQA | `MQA` |
|---|---|---|

```
MQA(x)
```

ARGUMENTE

| | |
|---|---|
| `x` | Daten als Vektor: Urliste oder Rangwertreihe<br>NA's zugelassen, werden vor Berechnung entfernt<br>Daten als diskrete Haeufigkeitstabelle<br>Daten als kontinuierliche Haeufigkeitstabelle |
| `SIL` | Arbeitmodus: = T lautlos, = F mit Information |

RESULT

MQA von x (gemäß entsprechender Formel)
NA bei unzulässigem Argument

CODE

siehe Seite 56

| `Spannweite` | Spannweite | `Spannweite` |
|---|---|---|

```
Spannweite(x)
```

ARGUMENTE

| | |
|---|---|
| `x` | Daten als Vektor: Urliste oder Rangwertreihe<br>NA's zugelassen, werden vor Berechnung entfernt |
| `SIL` | Arbeitmodus: = T lautlos, = F mit Information |

RESULT

max(x) - min(x)
NA bei unzulässigem Argument

CODE

siehe Seite 16

| `Stabdiagramm` | Stabdiagramm | `Stabdiagramm` |
|---|---|---|

`Stabdiagramm(x,MNY=F)`

ARGUMENTE

`x` Daten als Objekt “Diskrete Häufigkeitsverteilung

`MNY` legt Normierung in y-Richtung fest
MNY=T: Intervall [0,max(y)]
MNY=F: Intervall [0,1], dies ist Defaultsetzung

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Graph: Stabdiagramm

CODE

siehe Seite 23

| `Varianz` | Varianz | `Varianz` |
|---|---|---|

`Varianz(x)`

ARGUMENTE

`x` Daten als Vektor: Urliste oder Rangwertreihe
NA‘s zugelassen, werden vor Berechnung entfernt
Daten als diskrete Haeufigkeitstabelle
Daten als kontinuierliche Haeufigkeitstabelle

`SIL` Arbeitmodus: = T lautlos, = F mit Information

RESULT

Varianz von x (gemäß entsprechender Formel)
NA bei unzulässigem Argument

CODE

siehe Seite 53

# 10 Alphabetischer Index