

Package ‘robeth’

October 14, 2022

Type Package

Title R Functions for Robust Statistics

Version 2.7-6

Date 2020-03-02

Depends R (>= 3.2.0)

Author Alfio Marazzi <Alfio.Marazzi@unisante.ch>

Maintainer A. Randriamiharisoa <exelami@gmail.com>

Description Locations problems, M-estimates of coefficients and scale in linear regression, Weights for bounded influence regression, Covariance matrix of the coefficient estimates, Asymptotic relative efficiency of regression M-estimates, Robust testing in linear models, High breakdown point regression, M-estimates of covariance matrices, M-estimates for discrete generalized linear models.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2020-03-12 17:50:03 UTC

R topics documented:

robeth-package	5
addc	28
airef0	29
airefq	30
binprd	31
cerf	31
cerfd	32
cfrcov	32
Chi	33
chi	33

chisq	34
cia2b2	35
cibeat	35
cicloc	36
cifact	37
cimedv	37
cirock	38
comval	39
cquant	39
cyfalg	40
cygalg	41
cynalg	42
Dbinom	43
dfcomn	43
dfrpar	45
dfvals	45
dotp	46
dotpd	46
dpoiss	47
exch	48
exchd	48
fcum	49
Fn.Exp.f	50
fstord	50
gauss	51
gaussd	51
gfedca	52
gintac	53
glmdev	54
gyastp	55
gycstp	56
gymain	56
gytstp	58
h12	59
h12d	59
hymse	60
hylvse	61
hysest	62
hysestw	63
ingama	64
kfascv	64
kfedcb	65
kfedcc	66
kffacv	66
kiascv	67
kiedch	68
kiedcu	68
ktaskv	69

ktaskw	70
lgama	71
libet0	71
libeth	72
libetu	72
liclls	73
liepsh	73
liepsu	74
liindh	74
liinds	75
liindw	76
lilars	76
littst	77
lmdd	77
lrftcd	78
lyhalg	79
lyhdle	80
lymnwt	80
lytau2	81
lywalg	82
mchl	83
mchld	84
messagena	84
mff	85
mffd	85
mfragr	86
mfy	87
mfyd	87
mhat	88
minv	89
minvd	89
mirtsr	90
mly	91
mlyd	91
msf	92
msf1	93
msf1d	93
msfd	94
mss	94
mssd	95
mtt1	96
mtt1d	96
mtt2	97
mtt2d	97
mtt3	98
mtt3d	99
mty	99
mtyd	100

myhbhe	101
mymvlm	101
nlgm	102
nrm2	103
nrm2d	103
permc	104
permv	104
poissn	105
precd	106
prec	106
probst	107
Psi	107
psi	108
Psp	108
psp	109
QD2coef.f	109
QD2funC.f	110
Qn.Exp.f	111
quant	111
Random	112
Regtau.f	113
RegtauW.f	113
Rho	114
rho	115
ribet0	115
ribeth	116
ribetu	116
riclls	117
rilars	118
rimtrd	118
rimtrf	119
rmvc	120
ruben	120
rybifr	121
ryhalg	122
rynalg	123
rysalg	124
rysigm	125
rywalg	126
scal	127
scald	128
srt1	129
srt2	129
swap	130
swapd	131
tauare	132
tfrn2t	133
tftaut	133

tisrtc	134
to.character	135
to.double	135
to.integer	136
to.single	136
tquant	137
ttaskt	137
tteign	138
Ucv	139
ucv	139
ugl	140
Upcv	140
upcv	141
userfd	141
userfs	142
vcv	142
vpcv	143
Wcv	143
wcv	144
wfshat	144
wimedv	145
Wpcv	146
wpcv	146
Www	147
www	147
wyfalg	148
wyfcoll	149
wygalg	150
wynalg	151
xerf	152
xerp	152
xsy	153
xsyd	153
zemll	154

Index**155**

robeth-package *Interface for the FORTRAN programs developed at the ETH-Zuerich
and then at IUMSP-Lausanne*

Description

This package allows the computation of a broad class of procedures based on M-estimation and high breakdown point estimation, including robust regression, robust testing of linear hypotheses and robust covariances. The reference book quoted below is required for the theoretical background of the statistical and numerical methods

Details

Package: robeth
 Type: Package
 Version: 2.0
 Date: 2007-09-01
 License: GPL version 2 or later

Author(s)

Alfio Marazzi <Alfio.Marazzi@chuv.ch>

Maintainer: A. Randriamiharisoa <Alex.Randriamiharisoa@chuv.ch>

References

Marazzi A., (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California.

Examples

```
library(robeth)

#
# ----- Examples of Chapter 1: Location problems -----
#
y <- c(6.0,7.0,5.0,10.5,8.5,3.5,6.1,4.0,4.6,4.5,5.9,6.5)
n <- 12
dfvals()
#-----
# M-estimate (tm) of location and confidence interval (tl,tu)
#
dfrpar(as.matrix(y),"huber")
libeth()
s <- lilars(y); t0 <- s$theta; s0 <- s$sigma
s <- lyhalg(y=y,theta=t0,sigma=s0)
tm <- s$theta; var tm <- s$var
s <- quant(0.975)
tl <- tm-s*x*sqrt(vartm)
tu <- tm+s*x*sqrt(vartm)
#-----
# Hodges and Lehmann estimate (th) and confidence interval (zl,zu)
#
m <- n*(n+1)/2 # n even
k1 <- m/2; k2 <- k1+1
z1 <- lyhdle(y=y,k=k1); z2 <- lyhdle(y=y,k=k2)
th <- (z1$hdle+z2$hdle)/2.
ku <- liindh(0.95,n); k1 <- liindh(0.05,n)
zu <- lyhdle(y=y,k=ku$k); zl <- lyhdle(y=y,k=k1$k)
```

```

#.....
{
  cat(" tm, tl, tu : ",round(c(tm,tl,tu),3),"\n")
  cat(" th, zl, zu : ",round(c(th,zl$hdle,zu$hdle),3),"\n")
}
# tm, tl, tu : 5.809 4.748 6.87
# th, zl, zu : 5.85 5 7
#=====
#
# Two sample problem
#
y <- c(17.9,13.3,10.6,7.6,5.7,5.6,5.4,3.3,3.1,0.9)
n <- 10
x <- c(7.7,5.0,1.7,0.0,-3.0,-3.1,-10.5)
m <- 7
#-----
# Estimate (dm) and confidence interval [dl,du] based on Mann-Whitney
#
k1 <- m*n/2; k2 <- k1+1
s1 <- lymnwt(x=x,y=y,k=k1); s2 <- lymnwt(x=x,y=y,k=k2)
dm <- (s1$tmnwt+s2$tmnwt)/2.0
s1 <- liindw(0.05,m,n); k1 <- s1$k
s <- lymnwt(x=x,y=y,k=k1); dl <- s$tmnwt
s <- lymnwt(x=x,y=y,k=m*n-k1+1); du <- s$tmnwt
#-----
# Tau-test . P-value (P)
#
z <- c(x,y)
dfrpar(as.matrix(z),"huber")
libeth()
s <- lytau2(z=z,m=m,n=n)
P <- s$p
#
# estimate (tm) and confidence interval (tl,tu)
#
tm <- s$deltal
c22<- s$cov[3]
s <- quant(0.975)
tl <- tm-s*x*sqrt(c22)
tu <- tm+s*x*sqrt(c22)
#.....
{
  cat("dm, dl, du:",round(c(dm,dl,du),3),"\n")
  cat("P, tm, tl, tu:",round(c(P,tm,tl,tu),3),"\n")
}
# dm, dl, du: 6.35 2.9 12.9
# P, tm, tl, tu: 0.014 6.918 1.562 12.273

#
# Examples of Chapter 2: M-estimates of coefficients and scale in linear regression
#
# Read data; declare the vector wgt; load defaults
#

```

```

z <- c(-1, -2, 0, 35, 1, 0, -3, 20,
      -1, -2, 0, 30, 1, 0, -3, 39,
      -1, -2, 0, 24, 1, 0, -3, 16,
      -1, -2, 0, 37, 1, 0, -3, 27,
      -1, -2, 0, 28, 1, 0, -3, -12,
      -1, -2, 0, 73, 1, 0, -3, 2,
      -1, -2, 0, 31, 1, 0, -3, 31,
      -1, -2, 0, 21, 1, 0, -1, 26,
      -1, -2, 0, -5, 1, 0, -1, 60,
      -1, 0, 0, 62, 1, 0, -1, 48,
      -1, 0, 0, 67, 1, 0, -1, -8,
      -1, 0, 0, 95, 1, 0, -1, 46,
      -1, 0, 0, 62, 1, 0, -1, 77,
      -1, 0, 0, 54, 1, 0, 1, 57,
      -1, 0, 0, 56, 1, 0, 1, 89,
      -1, 0, 0, 48, 1, 0, 1, 103,
      -1, 0, 0, 70, 1, 0, 1, 129,
      -1, 0, 0, 94, 1, 0, 1, 139,
      -1, 0, 0, 42, 1, 0, 1, 128,
      -1, 2, 0, 116, 1, 0, 1, 89,
      -1, 2, 0, 105, 1, 0, 1, 86,
      -1, 2, 0, 91, 1, 0, 3, 140,
      -1, 2, 0, 94, 1, 0, 3, 133,
      -1, 2, 0, 130, 1, 0, 3, 142,
      -1, 2, 0, 79, 1, 0, 3, 118,
      -1, 2, 0, 120, 1, 0, 3, 137,
      -1, 2, 0, 124, 1, 0, 3, 84,
      -1, 2, 0, -8, 1, 0, 3, 101)
xx <- matrix(z,ncol=4, byrow=TRUE)
dimnames(xx) <- list(NULL,c("z2","xS","xT","y"))
z2 <- xx[,"z2"]; xS <- xx[,"xS"]; xT <- xx[,"xT"]
x <- cbind(1, z2, xS+xT, xS-xT, xS^2+xT^2, xS^2-xT^2, xT^3)
y <- xx[,"y"]
wgt <- vector("numeric",length(y))
n <- 56; np <- 7
dfvals()
# Set parameters for Huber estimate
dfrpar(x, "huber")
# Compute the constants beta, bet0, epsi2 and epsip
ribeth(wgt)
ribet0(wgt)
s <- liepsh()
epsi2 <- s$epsi2; epsip <- s$epsip
#
# Least squares solution (theta0,sigma0)
#
z <- riclls(x, y)
theta0<- z$theta; sigma0 <- z$sigma
# Preliminary estimate of the covariance matrix of the coefficients
cv <- kiascv(z$xt, fu=epsi2/epsip^2, fb=0.)
cov <- cv$cov
#-----
# Solution (theta1,sigma1) by means of RYHALG.

```



```

#
  zr  <- ryhalg(x,y,theta0,wgt,cov,sigmai=sigma0,ic=0)
  theta1<- zr$theta[1:np]; sigma1 <- zr$sigmaf; nit1 <- zr$nit
#-----
# Solution (theta2,sigma2) by means of RYWALG (recompute cov)
#
  cv <- ktaskv(x, f=epsi2/epsip^2)
  zr <- rywalg(x, y, theta0, wgt, cv$cov, sigmai=sigma0)
  theta2 <- zr$theta[1:np]; sigma2 <- zr$sigmaf; nit2 <- zr$nit
#-----
# Solution (theta3,sigma3) by means of RYSALG with ISIGMA=2.
#
  zr <- rysalg(x,y, theta0, wgt, cv$cov, sigma0, isigma=2)
  theta3 <- zr$theta[1:np]; sigma3 <- zr$sigmaf; nit3 <- zr$nit
#-----
# Solution (theta4,sigma4) by means of RYNALG with ICNV=2 and ISIGMA=0.
#
# Invert cov
  covm1 <- cv$cov
  zc <- mchl(covm1,np)
  zc <- minv(zc$a, np)
  zc <- mtt1(zc$r,np)
  covm1 <- zc$b
  zr <- rynalg(x,y, theta0, wgt, covm1, sigmai=sigma3,
              iopt=1, isigma=0, icnv=2)
  theta4 <- zr$theta[1:np]; sigma4 <- zr$sigmaf; nit4 <- zr$nit
#.....
{
  cat("theta0 : ",round(theta0[1:np],3),"\\n")
  cat("sigma0 : ",round(sigma0,3),"\\n")
  cat("theta1 : ",round(theta1,3),"\\n")
  cat("sigma1, nit1 : ",round(sigma1,3),nit1,"\\n")
  cat("theta2 : ",round(theta2,3),"\\n")
  cat("sigma2, nit2 : ",round(sigma2,3),nit2,"\\n")
  cat("theta3 : ",round(theta3,3),"\\n")
  cat("sigma3, nit3 : ",round(sigma3,3),nit3,"\\n")
  cat("theta4 : ",round(theta4,3),"\\n")
  cat("sigma4, nit4 : ",round(sigma4,3),nit4,"\\n")
}
# theta0 :  68.634  3.634  24.081 -8.053 -0.446 -0.179 -1.634
# sigma0 :  26.635
# theta1 :  70.006  5.006  24.742 -6.246 -0.079  0.434 -1.487
# sigma1, nit1 :  23.564  7
# theta2 :  70.006  5.006  24.742 -6.245 -0.079  0.434 -1.487
# sigma2, nit2 :  23.563  7
# theta3 :  69.993  5.002  24.766 -6.214 -0.055  0.44 -1.48
# sigma3, nit3 :  22.249  3
# theta4 :  69.993  5.002  24.766 -6.214 -0.055  0.44 -1.48
# sigma4, nit4 :  22.249  3

#
# ---- Examples of Chapter 3: Weights for bounded influence regression -----

```

```

#
#=====
rbmost <- function(x,y,cc,userfd) {
  n <- nrow(x); np <- ncol(x); dfcomn(xk=np)
  .dFvPut(1,"itw")
  z <- wimedv(x)
  z <- wyfalg(x, z$a, y, exu=userfd); nitw <- z$nit
  wgt <- 1/z$dist; wgt[wgt>1.e6] <- 1.e6
  z <- comval()
  bto <- z$bt0; ipso <- z$ipsi; co <- z$c
  z <- ribet0(wgt, itype=2, isqw=0)
  xt <- x*wgt; yt <- y * wgt
  z <- rilars(xt, yt)
  theta0 <- z$theta; sigma0 <- z$sigma
  rs <- z$rs/wgt; r1 <- rs/sigma0
  dfcomn(ipsi=1,c=cc)
  z <- liepsh(cc)
  den <- z$epsip
  g <- Psp(r1)/den # (see Psi in Chpt. 14)
  dfcomn(ipsi=ipso, c=co, bet0=bto)
  list(theta=theta0, sigma=sigma0, rs=rs, g=g, nitw=nitw)
}
#-----
# Mallows-standard estimate (with wyfalg and rywalg)
#
Mal.Std <- function(x, y, b2=-1, cc=-1, isigma=2) {
  n <- length(y); np <- ncol(x)
  dfrpar(x, "Mal-Std", b2, cc); .dFv <- .dFvGet()
  if (isigma==1) {dfcomn(d=.dFv$ccc); .dFvPut(1,"isg")}
# Weights
  z <- wimedv(x)
  z <- wyfalg(x, z$a, y); nitw <- z$nit
  wgt <- Www(z$dist) # See Www in Chpt. 14
# Initial cov. matrix of coefficient estimates
  z <- kiedch(wgt)
  cov <- ktaskw(x, z$d, z$e, f=1/n)
# Initial theta and sigma
  z <- rbmost(x,y,1.5,userfd)
  theta0 <- z$theta; sigma0 <- z$sigma; nitw0 <- z$nitw
# Final theta and sigma
  if (isigma==1) ribeth(wgt) else ribet0(wgt)
  z <- rywalg(x, y,theta0,wgt,cov$cov, sigmai=sigma0)
  theta1 <- z$theta[1:np]; sigma1 <- z$sigmaf; nit1 <- z$nit
# Covariance matrix of coefficient estimates
  z <- kfedcc(wgt, z$rs, sigma=sigma1)
  cov <- ktaskw(x, z$d, z$e, f=(sigma1^2)/n)
  sd1 <- NULL; for (i in 1:np) { j <- i*(i+1)/2
  sd1 <- c(sd1,cov$cov[j]) }
  sd1 <- sqrt(sd1)
#.....
{
  cat("rbmost: theta0 : ",round(theta0[1:np],3),"\\n")
}

```

```

cat("rbmost: sigma0, nitw : ",round(sigma0,3),nitw0,"\n")
cat("Mallows-Std: theta1 : ",round(theta1,3),"\n")
cat("Mallows-Std: sd1      : ",round(sd1,3),"\n")
cat("Mallows-Std: sigma1, nitw, nit1 : ",round(sigma1,3),nitw,nit1,"\n")
}

#.....
list(theta0=theta0[1:np], sigma0=sigma0, nitw=nitw,
      theta1=theta1, sigma1=sigma1, nit1=nit1, sd1=sd1)}
#-----
# Krasker-Welsch estimate (with wyalg and ryalg)
#
Kra.Wel <- function(x, y, ckw=-1, isigma=2) {
n <- length(y); np <- ncol(x)
dfrpar(x, "Kra-Wel", ckw); .dFv <- .dFvGet()
if (isigma==1) {dfcomn(d=.dFv$ccc); .dFvPut(1,"isg")}
# Weights
z <- wimedv(x)
z <- wyalg(x, z$a); nitw <- z$nit
wgt <- Www(z$dist) # See Www in Chpt. 14
# Initial cov. matrix of coefficient estimates
z <- kiedch(wgt)
cov <- ktaskw(x, z$d, z$e, f=1/n)
# Initial theta and sigma
z <- rbmost(x, y, cc=1.5)
theta0 <- z$theta; sigma0 <- z$sigma; nitw0 <- z$nitw
# Final theta and sigma
if (isigma==1) ribeth(wgt) else ribet0(wgt)
z <- ryalg(x, y,theta0,wgt,cov$cov, sigmai=sigma0)
theta2 <- z$theta[1:np]; sigma2 <- z$sigma; nit2 <- z$nit
#
# Covariance matrix of coefficient estimates
#
z <- kfedcc(wgt, z$rs, sigma=sigma2)
cov <- ktaskw(x, z$d, z$e, f=(sigma2^2)/n)
sd2 <- NULL; for (i in 1:np) { j <- i*(i+1)/2
sd2 <- c(sd2,cov$cov[j]) }
sd2 <- sqrt(sd2)
#.....
{
cat("rbmost: theta0 : ",round(theta0[1:np],3),"\n")
cat("rbmost: sigma0, nitw : ",round(sigma0,3),nitw0,"\n")
cat("Krasker-Welsch: theta2 : ",round(theta2,3),"\n")
cat("Krasker-Welsch: sd2      : ",round(sd2,3),"\n")
cat("Krasker-Welsch: sigma2, nitw, nit2 : ",round(sigma2,3),nitw,nit2,"\n")
}
#.....
list(theta0=theta0[1:np], sigma0=sigma0, nitw=nitw,
      theta2=theta2, sigma2=sigma2, nit2=nit2, sd2=sd2)}
#-----
# Read data; load defaults
#
z <- c( 8.2, 4, 23.005, 1, 7.6, 5, 23.873, 1,

```

```

      4.6, 0, 26.417, 1, 4.3, 1, 24.868, 1,
      5.9, 2, 29.895, 1, 5.0, 3, 24.200, 1,
      6.5, 4, 23.215, 1, 8.3, 5, 21.862, 1,
    10.1, 0, 22.274, 1, 13.2, 1, 23.830, 1,
    12.6, 2, 25.144, 1, 10.4, 3, 22.430, 1,
    10.8, 4, 21.785, 1, 13.1, 5, 22.380, 1,
    13.3, 0, 23.927, 1, 10.4, 1, 33.443, 1,
    10.5, 2, 24.859, 1, 7.7, 3, 22.686, 1,
    10.0, 0, 21.789, 1, 12.0, 1, 22.041, 1,
    12.1, 4, 21.033, 1, 13.6, 5, 21.005, 1,
    15.0, 0, 25.865, 1, 13.5, 1, 26.290, 1,
    11.5, 2, 22.932, 1, 12.0, 3, 21.313, 1,
    13.0, 4, 20.769, 1, 14.1, 5, 21.393, 1)
x   <- matrix(z, ncol=4, byrow=TRUE)
y   <- c( 7.6, 7.7, 4.3, 5.9, 5.0, 6.5, 8.3, 8.2, 13.2, 12.6,
        10.4, 10.8, 13.1, 12.3, 10.4, 10.5, 7.7, 9.5, 12.0, 12.6,
        13.6, 14.1, 13.5, 11.5, 12.0, 13.0, 14.1, 15.1)

dfvals()
dfcomn(xk=4)
cat("Results\n")
z1   <- Mal.Std(x, y)
z2   <- Kra.Wel(x, y)

#
# ---- Examples of Chapter 4: Covariance matrix of the coefficient estimates ----
#
#
# Read x[1:4] and then set x[,4] <- 1
#
z   <- c(80, 27, 89, 1, 80, 27, 88, 1, 75, 25, 90, 1,
        62, 24, 87, 1, 62, 22, 87, 1, 62, 23, 87, 1,
        62, 24, 93, 1, 62, 24, 93, 1, 58, 23, 87, 1,
        58, 18, 80, 1, 58, 18, 89, 1, 58, 17, 88, 1,
        58, 18, 82, 1, 58, 19, 93, 1, 50, 18, 89, 1,
        50, 18, 86, 1, 50, 19, 72, 1, 50, 19, 79, 1,
        50, 20, 80, 1, 56, 20, 82, 1, 70, 20, 91, 1)
x   <- matrix(z, ncol=4, byrow=TRUE)
n   <- 21; np <- 4; ncov <- np*(np+1)/2
dfvals()
# Cov. matrix of Huber-type estimates
dfrpar(x, "huber")
s   <- liepsh()
epsi2 <- s$epsi2; epsip <- s$epsip
z   <- rimtrf(x)
xt  <- z$x; sg <- z$sg; ip <- z$ip
zc  <- kiascv(xt, fu=epsi2/epsip^2, fb=0.)
covi <- zc$cov # Can be used in ryhalg with ic=0
zc  <- kfacv(xt, covi, f=1, sg=sg, ip=ip)
covf <- zc$cov
#.....
str <- rep(" ", ncov); str[cumsum(1:np)] <- "\n"

```

```
{
  cat("covf:\n")
  cat(round(covf,6),sep=str)
}

#
# ---- Examples of Chapter 5: Asymptotic relative efficiency ----
#
#-----
# Huber
#
  dfcomn(ipsi=1,c=1.345,d=1.345)
  .dFvPut(1,"ite")
  z <- airef0(mu=3, ialfa=1, sigmx=1)
#.....
{
  cat(" airef0 : Huber\n reff, alfa, beta, nit: ")
  cat(round(c(z$reff,z$alfa,z$beta,z$nit),3),sep=c(" ", " ", " ", " ", " ", "\n"))
}
#-----
# Schweppe: Krasker-Welsch
#
  dfcomn(ipsi=1,c=3.76,iucv=3,ckw=3.76,iwww=1)
  .dFvPut(3,"ite")
  z <- airef0(mu=3, ialfa=1, sigmx=1)
#.....
{
  cat(" airef0 : Krasker-Welsch\n reff, alfa, beta, nit: ")
  cat(round(c(z$reff,z$alfa,z$beta,z$nit),3),sep=c(" ", " ", " ", " ", " ", "\n"))
}
#-----
# Mallows-Standard
#
  dfcomn(ipsi=1,c=1.5,iucv=1,a2=0,b2=6.16,iww=3)
  .dFvPut(2,"ite")
  z <- airef0(mu=3, ialfa=1, sigmx=1)
#.....
{
  cat(" airef0 : Mallows-Std \n reff, alfa, beta, nit: ")
  cat(round(c(z$reff,z$alfa,z$beta,z$nit),3),sep=c(" ", " ", " ", " ", " ", "\n"))
}
#=====
z <- c(1, 0, 0,
      1, 0, 0,
      1, 0, 0,
      1, 0, 0,
      0, 1, 0,
      0, 1, 0,
      0, 1, 0,
      0, 1, 0,
      0, 1, 0,
      0, 0, 1,
      0, 0, 1,
```

```

      0, 0, 1,
      0, 0, 1)
tt <- matrix(z,ncol=3,byrow=TRUE)
n <- nrow(tt); mu <- 2
nu <- ncol(tt)

#-----
# Huber
#
  dfrpar(tt,"Huber")
  z <- airefq(tt, mu=mu, sigmx=1)
#.....
{
  cat(" airefq : Huber\n reff, beta, nit: ")
  cat(round(c(z$reff,z$beta,z$nit),3),sep=c(", ",", ", ",", ", ", "\n"))
}
#-----
# Krasker-Welsch
#
  dfrpar(tt,"kra-wel",upar=3.755)
  z <- airefq(tt, mu=mu, sigmx=1,init=1)
#.....
{
  cat(" airefq : Krasker-Welsch\n reff, beta, nit: ")
  cat(round(c(z$reff,z$beta,z$nit),3),sep=c(", ",", ", ", ", ", ", "\n"))
}
#-----
# Mallows Standard
#
  dfrpar(tt,"Mal-Std",1.1*(mu+nu),1.569)
  z <- airefq(tt, mu=mu, sigmx=1,init=1)
#.....
{
  cat(" airefq : Mallows-Std\n reff, beta, nit: ")
  cat(round(c(z$reff,z$beta,z$nit),3),sep=c(", ",", ", ", ", ", ", "\n"))
}

#
# ---- Examples of Chapter 6: Robust testing in linear models ----
#

#=====
tautest <- function(x,y,np,nq) {
# Full model. np variables in x[,1:np]
n <- nrow(x)
z <- riclls(x[,1:np], y)
theta0 <- z$theta; sigma0 <- z$sigma; .dFv <- .dFvGet()
z <- liepsh(.dFv$ccc) # ccc is globally defined by dfrpar
epsi2 <- z$epsi2; epsip <- z$epsip
zc <- ktaskv(x[,1:np],f=epsi2/(epsip^2))
covi <- zc$cov

```

```

ribeth(y)
zf    <- rywalg(x[,1:np],y,theta0,y,covi,sigma=sigma0)
thetaf <- zf$theta; sigma <- zf$sigmaf; rf <- zf$rs
f      <- kffacv(rf,np=np,sigma=sigma)
zc     <- ktaskv(x[,1:np],f=f$fh*(sigma^2)/n)
cov    <- zc$cov
# Sub-model: nq variables in x[,1:nq], nq < np
covi   <- cov[1:(nq*(nq+1)/2)]
zs     <- rywalg(x[,1:nq], y, thetaf, y, covi, sigma=sigma, isigma=0)
thetas <- zs$theta; rs <- zs$rs
# Compute Tau-test statistic and P-value
ztau   <- tftaut(rf,rs,y,rho,np,nq,sigma)
ftau   <- ztau$ftau
z       <- chisq(1,np-nq,ftau*epsip/epsi2)
P       <- 1-z$p
#.....
{
  cat(" F_tau, P, sigma: ")
  cat(round(c(ftau,P,sigma),3),sep=c(", ",", ",", "\n"))
  cat(" theta (small model): ",round(thetas[1:nq],3),"\n\n")
}
#.....
list(thetas=thetas[1:nq], sigma=sigma, rs=rs,ftau=ftau, P=P)}
#-----
dshift <- function(x, theta, sigma, rs, nq) {
# Shift estimate d and confidence interval
ncov   <- nq*(nq+1)/2
f       <- kffacv(rs,np=nq,sigma=sigma)
zc      <- ktaskv(x[,1:nq],f=f$fh)
cov     <- zc$cov
k11     <- 4.*cov[ncov-nq]
k12     <- 2.*cov[ncov-1]
k22     <- cov[ncov]
za      <- quant(0.05/2)
d       <- 2*theta[nq-1]/theta[nq]
q       <- za*x*sigma/theta[nq]
g       <- k22*(q^2)
a       <- d-g*k12/k22
b       <- abs(q)*sqrt(k11-2*d*k12+d*d*k22-g*(k11-k12*k12/k22))
dL      <- (a-b)/(1-g)
dU      <- (a+b)/(1-g)
#.....
cat(" d, dL, dU: ",round(c(d,dL,dU),3),sep=c(" ",", ",", ", ", ",", "\n"))
#.....
list(d=d, dL=dL, dU=dU)
}
#-----
potcy   <- function(m,ml,mu,h,k,d,cs,ct) {
fact    <- ((h*k) %% 2) + 1
r       <- exp(log(d)*(fact*m+h-k)/2 - log(ct/cs))
r1      <- exp(log(d)*(fact*ml+h-k)/2 - log(ct/cs))
ru      <- exp(log(d)*(fact*mu+h-k)/2 - log(ct/cs))
list(R=r, R1=r1, Ru=ru)}

```

```

#-----
rbmost <- function(x,y,cc,usext=userfd) {
  n      <- nrow(x); np <- ncol(x); dfcomn(xk=np)
  .dFvPut(1,"itw")
  z      <- wimedv(x)
  z      <- wyfalg(x, z$a, y, exu=usext); nitw <- z$nit
  wgt    <- 1/z$dist; wgt[wgt>1.e6] <- 1.e6
  z      <- comval()
  bto    <- z$bt0; ipso <- z$ipsi; co <- z$c
  z      <- ribet0(wgt, itype=2, isqw=0)
  xt     <- x*wgt; yt  <- y * wgt
  z      <- rilars(xt, yt)
  theta0 <- z$theta; sigma0 <- z$sigma
  rs     <- z$rs/wgt; r1    <- rs/sigma0
  dfcomn(ipsi=1,c=cc)
  z      <- liepsh(cc)
  den    <- z$epsip
  g      <- Psp(r1)/den # (see Psp in Chpt. 14)
  dfcomn(ipsi=ipso, c=co, bet0=bto)
  list(theta=theta0, sigma=sigma0, rs=rs, g=g, nitw=nitw)
}
#-----
dfvals()
z <- c(-1, -2, 0, 35, 1, 0, -3, 20,
      -1, -2, 0, 30, 1, 0, -3, 39,
      -1, -2, 0, 24, 1, 0, -3, 16,
      -1, -2, 0, 37, 1, 0, -3, 27,
      -1, -2, 0, 28, 1, 0, -3, -12,
      -1, -2, 0, 73, 1, 0, -3, 2,
      -1, -2, 0, 31, 1, 0, -3, 31,
      -1, -2, 0, 21, 1, 0, -1, 26,
      -1, -2, 0, -5, 1, 0, -1, 60,
      -1, 0, 0, 62, 1, 0, -1, 48,
      -1, 0, 0, 67, 1, 0, -1, -8,
      -1, 0, 0, 95, 1, 0, -1, 46,
      -1, 0, 0, 62, 1, 0, -1, 77,
      -1, 0, 0, 54, 1, 0, 1, 57,
      -1, 0, 0, 56, 1, 0, 1, 89,
      -1, 0, 0, 48, 1, 0, 1, 103,
      -1, 0, 0, 70, 1, 0, 1, 129,
      -1, 0, 0, 94, 1, 0, 1, 139,
      -1, 0, 0, 42, 1, 0, 1, 128,
      -1, 2, 0, 116, 1, 0, 1, 89,
      -1, 2, 0, 105, 1, 0, 1, 86,
      -1, 2, 0, 91, 1, 0, 3, 140,
      -1, 2, 0, 94, 1, 0, 3, 133,
      -1, 2, 0, 130, 1, 0, 3, 142,
      -1, 2, 0, 79, 1, 0, 3, 118,
      -1, 2, 0, 120, 1, 0, 3, 137,
      -1, 2, 0, 124, 1, 0, 3, 84,
      -1, 2, 0, -8, 1, 0, 3, 101)
xx <- matrix(z,ncol=4, byrow=TRUE)
dimnames(xx) <- list(NULL,c("z2","xS","xT","y"))

```



```

z2 <- xx[,"z2"]; xS <- xx[,"xS"]; xT <- xx[,"xT"]
x <- cbind(1, z2, xS+xT, xS-xT, xS^2+xT^2, xS^2-xT^2, xT^3)
y <- xx[,"y"]
z <- dfrpar(x, "huber",psipar=1.345)
#
# Tau-test and shift estimate
#
{
  cat("Results (linearity test)\n")
  np <- 7; nq <- 4 # Test linearity
  z <- tautest(x,y,np,nq)
  cat("Results (parallelism test)\n")
  np <- 4; nq <- 3 # Test parallelism
  z <- tautest(x,y,np,nq)
  z <- dshift(x, z$thetas, z$sigma, z$rs, nq)
}
#-----
# Input data; set defaults
#
z <- c(35.3, 20, 10.98,
      29.7, 20, 11.13,
      30.8, 23, 12.51,
      58.8, 20, 8.40,
      61.4, 21, 9.27,
      71.3, 22, 8.73,
      74.4, 11, 6.36,
      76.7, 23, 8.50,
      70.7, 21, 7.82,
      57.5, 20, 9.14,
      46.4, 20, 8.24,
      28.9, 21, 12.19,
      28.1, 21, 11.88,
      39.1, 19, 9.57,
      46.8, 23, 10.94,
      48.5, 20, 9.58,
      59.3, 22, 10.09,
      70.0, 22, 8.11,
      70.0, 11, 6.83,
      74.5, 23, 8.88,
      72.1, 20, 7.68,
      58.1, 21, 8.47,
      44.6, 20, 8.86,
      33.4, 20, 10.36,
      28.6, 22, 11.08)
x <- matrix(z, ncol=3, byrow=TRUE)
y <- x[,3]; x[,2:3] <- x[,1:2]; x[,1] <- 1
n <- length(y); np <- ncol(x); nq <- np - 1
#
# Optimal tau-test based on Scheppe-type estimates
#
z <- tauare(itype=3,mu=1,psi=2.665,bb=0,sigmax=1)
dfrpar(x, "Sch-Tau",upar=2.67); .dFvPut(1,"isg");
.dFv <- .dFvGet(); dfcomn(d=.dFv$ccc)

```

```

# Full model: initial estimates of theta, sigma and weights
dfcomn(xk=np) # Needed for userfd
zr    <- rbmost(x,y,cc=1.5)
theta0 <- zr$theta; sigma0 <- zr$sigma; nitw0 <- zr$nitw
#
# Initial and final values of weights
#
z      <- wimedv(x)
z      <- wyfalg(x, z$a, zr$g, nvarq=nq, igwt=1)
wgt    <- 1/z$dist ; wgt[wgt>1.e6] <- 1.e6
# Full model: covariance matrix of coefficients and inverse
z      <- kiedch(wgt)
zc     <- ktaskw(x, z$d, z$e, f=1/n, iainv=1)
cov    <- zc$cov; ainv <- zc$ainv
# Full model: Final estimate of theta and sigma
ribeth(wgt)
zf     <- rywalg(x, y, theta0, wgt, cov, sigmai=sigma0)
thetaf <- zf$theta[1:np]; sigma <- zf$sigmaf; nitf <- zf$nit
# Small model: Final estimate of theta and sigma
covi   <- cov[1:(nq*(nq+1)/2)]
xt     <- x[,1:nq,drop=FALSE]
zs     <- rywalg(xt, y, theta0, wgt, covi, sigmai=sigma, isigma=0)
thetas <- zs$theta[1:nq]; nits <- zs$nit
# Compute Tau-test statistic
ft     <- tftaut(zf$rs, zs$rs, wgt, rho, np, nq, sigma)
ftau   <- ft$ftau
# P-value
z      <- ttaskt(cov, ainv, np, nq, fact=n)
z      <- tteign(z$covtau, nq)
xlmbda <- z$xlmbda[1:(np-nq)]
mult   <- rep(1, length=np)
delta  <- rep(0, length=np)
z      <- ruben(xlmbda, delta, mult, ftau, xmode=-1, maxit=100, eps=0.0001)
P      <- 1-z$cumdf
#.....
{
  cat(" Optimal Tau-test : Schweppe-type estimates\n")
  cat(" theta0: ", round(theta0[1:np], 3), "\n sigma0, nitw: ")
  cat(round(c(sigma0, nitw0), 3), sep=c(" ", "\n"))
  cat(" thetáf: ", round(thetáf, 3), "\n sigma, nit: ")
  cat(round(c(sigma, nitf), 3), sep=c(" ", "\n"))
  cat(" thetas: ", round(thetas, 3), "\n sigma, nit: ")
  cat(round(c(sigma, nits), 3), sep=c(" ", "\n"))
  cat(" F_tau = ", round(ftau, 3), ", P = ", P, "\n")
}
#=====
rn2mal <- function(x,y,b2,c,nq) {
  n <- length(y); np <- ncol(x)
  #
  # Rn2-test on Mallows estimators
  # =====
  dfrpar(x, "mal-std", b2, c)
  .dFvPut(1, "isg"); .dFv <- .dFvGet(); dfcomn(d=.dFv$ccc)

```

```

#
# Initial and final values of weights
#
z     <- wimedv(x)
z     <- wyfalg(x, z$a, wgt)
wgt   <- Www(z$dist); nitw <- z$nit
#
# Initial theta and sigma (using weighted LAR)
#
ribet0(wgt, isqw=0)
xt    <- x*wgt
yt    <- y * wgt
z     <- rilars(xt, yt)
theta0 <- z$theta; sigma0 <- z$sigma
#
# Initial value of COV
#
z     <- kiedch(wgt)
zc    <- ktaskw(x, z$d, z$e, f=1/n)
covi  <- zc$cov
#
# Solution by means of RYWALG.
#
z     <- ribeth(wgt)
beta  <- z$bta
zw    <- rywalg(x, y, theta0, wgt, covi, sigma0)
theta1 <- zw$theta[1:np]; sigma1 <- zw$sigmaf; nit1 <- zw$nit
#
# Unscaled covariance matrix of coefficients
#
zc    <- kfedcb(wgt, zw$rs, sigma=sigma1)
z     <- ktaskw(x, zc$d, zc$e, f=1/n)
cov1  <- zc$cov
#
# Rn2-test statistic and significance
#
z     <- tfrn2t(cov1, theta1, n, nq)
rn2m  <- z$rn2t/(n*sigma1^2)
z     <- chisq(1, np-nq, rn2m)
p1    <- 1.-z$p
list(theta1=theta1, sigma1=sigma1, wgt=wgt, nitw=nitw, nit1=nit1,
      rn2m=rn2m, p1=p1)}
#-----
#
# Read data
#
z <- c(35.3, 20, 10.98,
       29.7, 20, 11.13,
       30.8, 23, 12.51,
       58.8, 20, 8.40,
       61.4, 21, 9.27,
       71.3, 22, 8.73,
       74.4, 11, 6.36,

```

```

76.7, 23, 8.50,
70.7, 21, 7.82,
57.5, 20, 9.14,
46.4, 20, 8.24,
28.9, 21, 12.19,
28.1, 21, 11.88,
39.1, 19, 9.57,
46.8, 23, 10.94,
48.5, 20, 9.58,
59.3, 22, 10.09,
70.0, 22, 8.11,
70.0, 11, 6.83,
74.5, 23, 8.88,
72.1, 20, 7.68,
58.1, 21, 8.47,
44.6, 20, 8.86,
33.4, 20, 10.36,
28.6, 22, 11.08)
x <- matrix(z, ncol=3, byrow=TRUE)
y <- x[,3]; x[,2:3] <- x[,1:2]; x[,1] <- 1
n <- length(y); np <- ncol(x); nq <- np - 1
wgt <- vector("numeric",length(y))
z <- rn2mal(x, y, 4, 1.5, nq)
#.....
{
cat("Rn2-test on Mallows estimators\n")
cat(" theta1: ",round(z$theta1,3),"\n sigma1, nitw, nit1: ")
cat(round(c(z$sigma1,z$nitw,z$nit1),3),sep=c(" ", " ", " ", "\n"))
cat(" Rn2 =",round(z$rn2m,3),", P =",z$p1,"\n")
}

#
# ---- Examples of Chapter 7: Breakdown point regression ----
#
#
# Read data; load defaults
#
z <- c(80, 27, 89, 42,
      80, 27, 88, 37,
      75, 25, 90, 37,
      62, 24, 87, 28,
      62, 22, 87, 18,
      62, 23, 87, 18,
      62, 24, 93, 19,
      62, 24, 93, 20,
      58, 23, 87, 15,
      58, 18, 80, 14,
      58, 18, 89, 14,
      58, 17, 88, 13,
      58, 18, 82, 11,
      58, 19, 93, 12,
```

```

      50, 18, 89,  8,
      50, 18, 86,  7,
      50, 19, 72,  8,
      50, 19, 79,  8,
      50, 20, 80,  9,
      56, 20, 82, 15,
      70, 20, 91, 15)
x     <- matrix(z, ncol=4, byrow=TRUE)
y     <- x[,4]; x[,4] <- 1
n     <- length(y); np <- ncol(x)
nq    <- np+1
dfvals()
#
# Least median of squares
#
zr    <- hylmse(x,y, nq, ik=1, iopt=3, intch=1)
theta1 <- zr$theta; xmin1 <- zr$xmin
zr    <- hylmse(x,y, nq, ik=2, iopt=3, intch=1)
theta2 <- zr$theta; xmin2 <- zr$xmin
zr    <- hylmse(x,y, nq, ik=1, iopt=1, intch=1)
theta3 <- zr$theta; xmin3 <- zr$xmin

#
# Least trimmed squares
#
zr    <- hyltse(x,y, nq, ik=1, iopt=3, intch=1)
theta4 <- zr$theta; xmin4 <- zr$smin
zr    <- hyltse(x,y, nq, ik=2, iopt=3, intch=1)
theta5 <- zr$theta; xmin5 <- zr$smin
zr    <- hyltse(x,y, nq, ik=1, iopt=1, intch=1)
theta6 <- zr$theta; xmin6 <- zr$smin

#
# S-estimate
#
z     <- dfrpar(x,'S')
z     <- ribetu(y)
zr    <- hysest(x,y, nq, iopt=3, intch=1)
theta7 <- zr$theta[1:np]; xmin7 <- zr$smin
zr    <- hysest(x,y, nq, iopt=1, intch=1)
theta8 <- zr$theta[1:np]; xmin8 <- zr$smin
#.....
{
  cat("Results\n theta1 = (")
  cat(round(theta1,3),sep=" ",")
  cat("), xmin1 ="); cat(round(xmin1,3))
  cat("\n theta2 = ("); cat(round(theta2,3),sep=" ",")
  cat("), xmin2 ="); cat(round(xmin2,3))
  cat("\n theta3 = ("); cat(round(theta3,3),sep=" ",")
  cat("), xmin3 ="); cat(round(xmin3,3))
  cat("\n theta4 = ("); cat(round(theta4,3),sep=" ",")
  cat("), xmin4 ="); cat(round(xmin4,3))
  cat("\n theta5 = ("); cat(round(theta5,3),sep=" ",")

```

```

cat("", xmin5 ="); cat(round(xmin5,3))
cat("\n theta6 = ("); cat(round(theta6,3),sep=" ")
cat("", xmin6 ="); cat(round(xmin6,3))
cat("\n theta7 = ("); cat(round(theta7,3),sep=" ")
cat("", xmin7 ="); cat(round(xmin7,3))
cat("\n theta8 = ("); cat(round(theta8,3),sep=" ")
cat("", xmin8 ="); cat(round(xmin8,3),"\n")
}

#
# ---- Examples of Chapter 8: M-estimates of covariance matrices ----
#
#
# Read data; set defaults
#
z <- c(4.37, 5.23, 4.38, 5.02,
      4.56, 5.74, 4.42, 4.66,
      4.26, 4.93, 4.29, 4.66,
      4.56, 5.74, 4.38, 4.90,
      4.30, 5.19, 4.22, 4.39,
      4.46, 5.46, 3.48, 6.05,
      3.84, 4.65, 4.38, 4.42,
      4.57, 5.27, 4.56, 5.10,
      4.26, 5.57, 4.45, 5.22,
      4.37, 5.12, 3.49, 6.29,
      3.49, 5.73, 4.23, 4.34,
      4.43, 5.45, 4.62, 5.62,
      4.48, 5.42, 4.53, 5.10,
      4.01, 4.05, 4.45, 5.22,
      4.29, 4.26, 4.53, 5.18,
      4.42, 4.58, 4.43, 5.57,
      4.23, 3.94, 4.38, 4.62,
      4.42, 4.18, 4.45, 5.06,
      4.23, 4.18, 4.50, 5.34,
      3.49, 5.89, 4.45, 5.34,
      4.29, 4.38, 4.55, 5.54,
      4.29, 4.22, 4.45, 4.98,
      4.42, 4.42, 4.42, 4.50,
      4.49, 4.85)
cx <- matrix(z, ncol=2, byrow=TRUE)
n <- nrow(cx); np <- ncol(cx)
dst0 <- vector("numeric",n)
#-----
# Classical covariance
#
t0 <- apply(cx, 2, mean)
xmb <- sweep(cx, 2, t0)
cv0 <- crossprod(xmb)/n
# Mahalanobis distances
cvm1 <- solve(cv0)
for (i in 1:n) {
  z <- xmb[i,,drop=FALSE]; dst0[i] <- sqrt(z %*% cvm1 %*% t(z))}

```

```

=====
# M-estimate of covariance
#
  zc  <- cicloc()
  za  <- cia2b2(nvar=np)
  a2  <- za$a2; b2 <- za$b2
  zd  <- cibeat(a2, b2, np)
  cw  <- zc$c;  dv <- zd$d
  dfcomn(iucv=1, a2=a2, b2=b2, bt=dv, cw=cw)
#  zf  <- cifact(a2,b2,np); fc <- zf$fc
  z   <- cimedv(cx)
  ai  <- z$a; ti <- z$t; fc <- 1
#-----
# With prescription F0
  zd  <- cyfalg(cx,ai,ti)
  zc  <- cfrcov(zd$a,np,fc)
  cv1 <- zc$cov; t1 <- zd$t; dst1 <- zd$dist; nt1 <- zd$nit
#-----
# With prescription NH
  zd  <- cynalg(cx,ai,ti)
  zc  <- cfrcov(zd$a,np,fc)
  cv2 <- zc$cov; t2 <- zd$t; dst2 <- zd$dist; nt2 <- zd$nit
#-----
# With prescription CG
  zd  <- cygalg(cx,ai,ti)
  zc  <- cfrcov(zd$a,np,fc)
  cv3 <- zc$cov; t3 <- zd$t; dst3 <- zd$dist; nt3 <- zd$nit
#.....
{
  cat("Results\n\n cv0[1,1],cv0[2,1],cv0[2,2] = (")
  cat(round(as.vector(cv0)[-2],3),sep=", ")
  cat(")\n t0 = ("); cat(round(t0,3),sep=", ")
  cat(")\n dist0 :\n ")
  cat(round(dst0,3),sep=c(rep(", ",9)," ,\n "))
  cat("\n cv1[1,1],cv1[2,1],cv1[2,2] = (")
  cat(round(cv1,3),sep=", ")
  cat(")\n t1 = ("); cat(round(t1,3),sep=", ")
  cat("), nit1 =",nt1); cat("\n dist1 :\n ")
  cat(round(dst1,3),sep=c(rep(", ",9)," ,\n "))
  cat("\n cv2[1,1],cv2[2,1],cv2[2,2] = (")
  cat(round(cv2,3),sep=", ")
  cat(")\n t2 = ("); cat(round(t2,3),sep=", ")
  cat("), nit2 =",nt2); cat("\n dist2 :\n ")
  cat(round(dst2,3),sep=c(rep(", ",9)," ,\n "))
  cat("\n cv3[1,1],cv3[2,1],cv3[2,2] = (")
  cat(round(cv3,3),sep=", ")
  cat(")\n t3 = ("); cat(round(t3,3),sep=", ")
  cat("), nit3 =",nt3); cat("\n dist3 :\n ")
  cat(round(dst3,3),sep=c(rep(", ",9)," ,\n "))
}

```

```

#
# ----- Examples of Chapter 9: Mixed procedures -----
#

bindec <- function(np,ind,cpc,cpr) {
  n <- length(ind)
  ccar <- matrix("-",ncol=np, nrow=n)
  for (i in 1:n) {
    j <- 0
    num <- abs(ind[i])
    while (num != 0 & j < np) {
      j <- j+1
      if (num %% 2 == 1) ccar[i,j] <- "X"
      num <- num %% 2}}
    data.frame(Cp=round(cpc,3),Cp.r=round(cpr,3),ipr=ind,i=ccar)
  }
}
#-----
# Read data
#
z <- c(-1, -2, 0, 35, 1, 0, -3, 20,
      -1, -2, 0, 30, 1, 0, -3, 39,
      -1, -2, 0, 24, 1, 0, -3, 16,
      -1, -2, 0, 37, 1, 0, -3, 27,
      -1, -2, 0, 28, 1, 0, -3, -12,
      -1, -2, 0, 73, 1, 0, -3, 2,
      -1, -2, 0, 31, 1, 0, -3, 31,
      -1, -2, 0, 21, 1, 0, -1, 26,
      -1, -2, 0, -5, 1, 0, -1, 60,
      -1, 0, 0, 62, 1, 0, -1, 48,
      -1, 0, 0, 67, 1, 0, -1, -8,
      -1, 0, 0, 95, 1, 0, -1, 46,
      -1, 0, 0, 62, 1, 0, -1, 77,
      -1, 0, 0, 54, 1, 0, 1, 57,
      -1, 0, 0, 56, 1, 0, 1, 89,
      -1, 0, 0, 48, 1, 0, 1, 103,
      -1, 0, 0, 70, 1, 0, 1, 129,
      -1, 0, 0, 94, 1, 0, 1, 139,
      -1, 0, 0, 42, 1, 0, 1, 128,
      -1, 2, 0, 116, 1, 0, 1, 89,
      -1, 2, 0, 105, 1, 0, 1, 86,
      -1, 2, 0, 91, 1, 0, 3, 140,
      -1, 2, 0, 94, 1, 0, 3, 133,
      -1, 2, 0, 130, 1, 0, 3, 142,
      -1, 2, 0, 79, 1, 0, 3, 118,
      -1, 2, 0, 120, 1, 0, 3, 137,
      -1, 2, 0, 124, 1, 0, 3, 84,
      -1, 2, 0, -8, 1, 0, 3, 101)
xx <- matrix(z,ncol=4, byrow=TRUE)
dimnames(xx) <- list(NULL,c("z2","xS","xT","y"))
z2 <- xx[,"z2"]; xS <- xx[,"xS"]; xT <- xx[,"xT"]
x <- cbind(1, z2, xS+xT, xS-xT, xS^2+xT^2, xS^2-xT^2, xT^3)
y <- xx[,"y"]

```



```

    wgt  <- vector("numeric",length(y))
    n    <- 56; np <- 7
dfvals()
# Compute classical sigma and the t-statistics
dfrpar(x,"ols",-1,-1); .dFv  <- .dFvGet()
z      <- mirtsr(x,y,.dFv$ite)
sigmc  <- z$sigma; tstac <- z$t

# Compute robust sigma and the t-statistics
dfrpar(x,"huber",-1,-1); .dFv  <- .dFvGet()
z      <- mirtsr(x,y,.dFv$ite)
sigmr  <- z$sigma; tstar <- z$t
#
# All possible regressions including the constant and linear terms
#
vp      <- rep(-0.5, length=np)
vp[1]   <- 3; vp[3] <- 2; vp[4] <- 1
za      <- mfragr(x, y, vp, nc=18, .dFv$ite, sigmac=sigmc, sigmar=sigmr)
#
# Priorities by means of t-directed search
#
zt      <- mfragr(x, y, tstar, nc=7, .dFv$ite, sigmac=sigmc, sigmar=sigmr)
#.....
{
cat(" Estimates of sigma\n ")
cat(" sigmc =",round(sigmc,3)," , sigmr =",round(sigmr,3)," \n")
cat(" Regressions on subset of variables:\n")
cat(" C{p} C{p,@} ipr 1 2 3 4 5 6 7\n")
cat(t(bindec(np,za$ipr,za$cpc,za$cpr)),sep=c(rep(" ",9)," \n"))
cat("\n t-directed search\n")
cat(" tstar[1:7]=(", round(tstar,3),sep=c(" ",rep(" ",6)))
cat(")\n C_p C{p,@} ipr 1 2 3 4 5 6 7\n")
cat(t(bindec(np,zt$ipr,zt$cpc,zt$cpr)),sep=c(rep(" ",9)," \n"))
}
#=====
#
# Read data; set defaults
#
z <- c(4.37, 5.23, 4.48, 5.42, 4.38, 5.02, 4.53, 5.10,
4.56, 5.74, 4.01, 4.05, 4.42, 4.66, 4.45, 5.22,
4.26, 4.93, 4.29, 4.26, 4.29, 4.66, 4.53, 5.18,
4.56, 5.74, 4.42, 4.58, 4.38, 4.90, 4.43, 5.57,
4.30, 5.19, 4.23, 3.94, 4.22, 4.39, 4.38, 4.62,
4.46, 5.46, 4.42, 4.18, 3.48, 6.05, 4.45, 5.06,
3.84, 4.65, 4.23, 4.18, 4.38, 4.42, 4.50, 5.34,
4.57, 5.27, 3.49, 5.89, 4.56, 5.10, 4.45, 5.34,
4.26, 5.57, 4.29, 4.38, 4.45, 5.22, 4.55, 5.54,
4.37, 5.12, 4.29, 4.22, 3.49, 6.29, 4.45, 4.98,
3.49, 5.73, 4.42, 4.42, 4.23, 4.34, 4.42, 4.50,
4.43, 5.45, 4.49, 4.85, 4.62, 5.62)
cx    <- matrix(z, ncol=2, byrow=TRUE)
n     <- nrow(cx); np <- ncol(cx)
y     <- vector("numeric",length=n)

```

```

#
# Minimum Volume Ellipsoid covariances
#
dfvals(); .dFv <- .dFvGet()
z <- mymvlm(cx,y,ilms=0,iopt=3,iseed=5321)
dst <- z$d; cv <- z$cov; xvol <- z$xvol
#.....
{
  cat("Minimum Volume Ellipsoid covariances\n cv = (")
  cat(round(cv,3),sep=c(" ", " ", " "))
  cat("), Objective function value =",round(xvol,3),"\ndistances:\n")
  cat(round(dst,3),sep=c(rep(" ",9),"",\n"))
}
#=====
#
# Read data; load defaults
#
z <- c(80, 27, 89, 42,
      80, 27, 88, 37,
      75, 25, 90, 37,
      62, 24, 87, 28,
      62, 22, 87, 18,
      62, 23, 87, 18,
      62, 24, 93, 19,
      62, 24, 93, 20,
      58, 23, 87, 15,
      58, 18, 80, 14,
      58, 18, 89, 14,
      58, 17, 88, 13,
      58, 18, 82, 11,
      58, 19, 93, 12,
      50, 18, 89, 8,
      50, 18, 86, 7,
      50, 19, 72, 8,
      50, 19, 79, 8,
      50, 20, 80, 9,
      56, 20, 82, 15,
      70, 20, 91, 15)
x <- matrix(z, ncol=4, byrow=TRUE)
y <- x[,4]; x[,4] <- 1
n <- length(y); np <- ncol(x)
nq <- np+1
dfvals()
#
# High breakdown point & high efficiency regression
#
dfrpar(x,"S",-1,-1)
z <- myhbhe(x, y, iseed=5431)
#.....
{
  cat("High breakdown point & high efficiency regression\n")
  cat(" theta0 = ("); cat(round(z$theta0,3),sep=" ", " ")
  cat("), sigma0 =",round(z$sigm0,3))
}

```

```

cat("\n theta1 = ("); cat(round(z$theta1,3),sep=", ")
cat("), sigma1 = ",round(z$sigm1,3),", ",tbias =",sep="")
cat(round(z$tbias,3),"\n")
}

#
# ----- Examples of Chapter 10: M-estimates for discrete GLM -----
#

glmb <- function(x,y,n,np,upar) {
#
# BI estimates of theta, A, ci and wa: Bernouilli responses, b=upar
#
# Initial theta, A (A0) and c (c0)
#
ni <- rep.int(1,n)
z <- gintac(x,y,ni,icase=1,b=upar,c=1.5)
theta0 <- z$theta[1:np]; A0 <- z$a; c0 <- z$ci
# Initial distances |Ax_i| and cut off points a_i (wa)
wa <- upar/z$dist
vtheta <- x %*% theta0
z <- gfedca(vtheta, c0, wa, ni, icase=1)
zc <- ktaskw(x, z$d, z$e, f=1/n) # See Chpt. 4
covi <- z$cov
# Final theta, A, c (ci) and a (wa)
z <- gyman(x, y, ni, covi, A0, theta0, b = upar)
theta <- z$theta; A <- z$a; ci <- z$ci; wa <- z$wa; nit <- z$nit
# Final cov. matrix and std. dev's of coeff. estimates
z <- gfedca(z$vtheta, ci, wa, ni, icase=1)
sdev <- NULL
zc <- ktaskw(x, z$d, z$e, f=1/n)
for (i in 1:np) {ii <- i*(i+1)/2; sdev <- c(sdev,z$cov[ii])}
sdev <- sqrt(sdev)
list(theta=theta, A=A, ci=ci, wa=wa, nit=nit, sdev=sdev)}
#-----
# Read data; load defaults
#
z <- c(3.70, 0.825, 1, 3.50, 1.090, 1,
1.25, 2.500, 1, 0.75, 1.500, 1,
0.80, 3.200, 1, 0.70, 3.500, 1,
0.60, 0.750, 0, 1.10, 1.700, 0,
0.90, 0.750, 0, 0.90, 0.450, 0,
0.80, 0.570, 0, 0.55, 2.750, 0,
0.60, 3.000, 0, 1.40, 2.330, 1,
0.75, 3.750, 1, 2.30, 1.640, 1,
3.20, 1.600, 1, 0.85, 1.415, 1,
1.70, 1.060, 0, 1.80, 1.800, 1,
0.40, 2.000, 0, 0.95, 1.360, 0,
1.35, 1.350, 0, 1.50, 1.360, 0,
1.60, 1.780, 1, 0.60, 1.500, 0,
1.80, 1.500, 1, 0.95, 1.900, 0,
1.90, 0.950, 1, 1.60, 0.400, 0,

```

```

      2.70, 0.750, 1,   2.35, 0.300, 0,
      1.10, 1.830, 0,   1.10, 2.200, 1,
      1.20, 2.000, 1,   0.80, 3.330, 1,
      0.95, 1.900, 0,   0.75, 1.900, 0,
      1.30, 1.625, 1)
x   <- matrix(z, ncol=3, byrow=TRUE)
y   <- x[,3]; x[,3] <- log(x[,2]); x[,2] <- log(x[,1]) ; x[,1] <- 1
n   <- length(y); np <- ncol(x)
dfvals()
upar <- 3.2*sqrt(np)
z1   <- glmb(x,y,n,np,upar)

upar <- 3.7*sqrt(np)
z2   <- glmb(x,y,n,np,upar)

z     <- glmb(x,y,n,np,300) # Classical estimates
#.....
{
cat("\n Robust estimates : upar=5.5426, nit =",z1$nit,"\n")
cat(" {theta[i] (sdev[i]), i=1:3}\n ")
for (i in 1:3) cat(round(z1$theta[i],3)," (",round(z1$sdev[i],3)," ",sep="")
cat("\n\n Robust estimates : upar=6.4086, nit =",z2$nit,"\n")
cat(" {theta[i] (sdev[i]), i=1:3}\n ")
for (i in 1:3) cat(round(z2$theta[i],3)," (",round(z2$sdev[i],3)," ",sep="")
cat("\n\n Classical estimates : upar=300, nit =",z$nit,"\n")
cat(" {theta[i] (sdev[i]), i=1:3}\n ")
for (i in 1:3) cat(round(z$theta[i],3)," (",round(z$sdev[i],3)," ",sep="")
cat("\n")
}
#-----

```

addc	<i>Adds a column vector to a transformed design matrix and updates its QR-decomposition</i>
------	---

Description

See Marazzi A. (1993), p.355

Usage

```
addc(x, n = nrow(x), l, j, ip)
```

Arguments

x	See reference
n	See reference
l	See reference

j See reference
ip See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.355

airef0	<i>Asymptotic relative efficiency of a general M-estimate for a model with mu quantitative covariates with or without a constant term</i>
--------	---

Description

See Marazzi A. (1993), p.167

Usage

```
airef0(expsi = psi, exu = ucv, exw = www, itype = .dFvGet()$ite, mu,
       ialfa = .dFvGet()$ial, sigmx = 1, upper = .dFvGet()$upr,
       til = .dFvGet()$tli, maxit = .dFvGet()$mxe, tol = .dFvGet()$tlo)
```

Arguments

expsi See reference
exu See reference
exw See reference
itype See reference
mu See reference
ialfa See reference
sigmx See reference
upper See reference
til See reference
maxit See reference
tol See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.167

airefq	<i>Asymptotic relative efficiency of a general M-estimate for a model with mu quantitative and nu qualitative covariates</i>
--------	--

Description

See Marazzi A. (1993), p.170

Usage

```
airefq(t, expsi = psi, exu = ucv, exw = www, itype = .dFvGet()$ite, mu, sigmx = 1,
      upper = .dFvGet()$upr, til = .dFvGet()$tli, tau = .dFvGet()$tua, nobs = nrow(t),
      maxit = .dFvGet()$mxe, tol = .dFvGet()$tlo, init = .dFvGet()$ini,
      nitmon = .dFvGet()$ntm)
```

Arguments

t	See reference
expsi	See reference
exu	See reference
exw	See reference
itype	See reference
mu	See reference
sigmx	See reference
upper	See reference
til	See reference
tau	See reference
nobs	See reference
maxit	See reference
tol	See reference
init	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.170

binprd	<i>Binomial probability distribution</i>
--------	--

Description

See Marazzi A. (1993), p.367

Usage

binprd(k, n, p)

Arguments

k	See reference
n	See reference
p	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.367

cerf	<i>Complemented error function (single precision)</i>
------	---

Description

See Marazzi A. (1993), p.380

Usage

cerf(x)

Arguments

x	See reference
---	---------------

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.380

cerfd	<i>Complemented error function (double precision)</i>
-------	---

Description

See Marazzi A. (1993), p.380

Usage

cerfd(x)

Arguments

x See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.380

cfrcov	<i>Computation of $fC.fC.inv(AT A)$ for a given matrix A and scale factor fC</i>
--------	---

Description

See Marazzi A. (1993), p.242

Usage

cfrcov(a, nvar, fc, tau = .dFvGet())\$tua)

Arguments

a See reference
nvar See reference
fc See reference
tau See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.242

Chi	<i>Chi weight function for location and regression</i>
-----	--

Description

See Marazzi A. (1993), p.322

Usage

Chi(svals)

Arguments

svals A vector of input values

Value

The values of the chi function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.32

chi	<i>Chi weight function for location and regression</i>
-----	--

Description

See Marazzi A. (1993), p.322

Usage

chi(s)

Arguments

s A scalar input value

Value

The value of the chi function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.322

chisq

Cumulative Chi-square distribution function

Description

See Marazzi A. (1993), p.373

Usage

chisq(kode = 1, ifn, x)

Arguments

kode	See reference
ifn	See reference
x	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.373

cia2b2	<i>Determination of the parameters a2 and b2 of the Huber weight function from the proportion eps of contamination</i>
--------	--

Description

See Marazzi A. (1993), p.244

Usage

```
cia2b2(eps = .dFvGet()$esp, nvar, tol = .dFvGet()$tlo, maxit = .dFvGet()$mxt)
```

Arguments

eps	See reference
nvar	See reference
tol	See reference
maxit	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.244

cibeat	<i>Determination of the parameter d of the Huber weight function</i>
--------	--

Description

See Marazzi A. (1993), p.247

Usage

```
cibeat(a2 = .dFvGet()$aa2, b2 = .dFvGet()$bb2, nvar)
```

Arguments

a2	See reference
b2	See reference
nvar	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.247

cicloc

Determination of the parameter c of the Huber weight function from the proportion eps of contamination

Description

See Marazzi A. (1993), p.243

Usage

```
cicloc(eps = .dFvGet())$esp, tol = .dFvGet())$tlo)
```

Arguments

eps See reference
tol See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.243

cifact	<i>Determination of the correction factor for the M-estimate based on Huber weight function</i>
--------	---

Description

See Marazzi A. (1993), p.246

Usage

```
cifact(a2 = .dFvGet()$aa2, b2 = .dFvGet()$bb2, nvar, tol = .dFvGet()$tlo,
       maxit = .dFvGet()$mxt)
```

Arguments

a2	See reference
b2	See reference
nvar	See reference
tol	See reference
maxit	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.246

cimedv	<i>Initial values for the iterative algorithms implemented in CYFALG, CYNALG, and CYGALG</i>
--------	--

Description

See Marazzi A. (1993), p.230

Usage

```
cimedv(x, nobs = nrow(x), nfirst = nobs, iloc = .dFvGet()$ilc, t)
```

Arguments

x	See reference
nobs	See reference
nfirst	See reference
iloc	See reference
t	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.230

cirock	<i>Initial values for the Rocke estimates of covariance</i>
--------	---

Description

See Marazzi A. (1993), p.223

Usage

```
cirock(nvar, em = .dFvGet()$em, cr = .dFvGet()$cr, iopt = 1)
```

Arguments

nvar	See the description of nvar as indicated above
em	See the description of em as indicated above
cr	See the description of cr as indicated above
iopt	See the description of iopt as indicated above

Author(s)

Rocke and Downs (1981)

References

<https://www.ubc.ca/search/?q=rocke#gsc.tab=0&gsc.q=rocke&gsc.page=1> - Marazzi A. (1993), *Algorithm, Routines, and S functions for Robust Statistics*, Wadsworth & Brooks/cole, Pacific Grove, California, p.223

comval	<i>Gives the current values of the parameters of the ROBETH subroutine common blocks</i>
--------	--

Description

See Marazzi A. (1993), p.405

Usage

comval()

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.405

cquant	<i>Inverse of the cumulative Chi2-distribution function</i>
--------	---

Description

See Marazzi A. (1993), p.374

Usage

cquant(p, ifn, tol = 5e-06, maxit = 50)

Arguments

p	See reference
ifn	See reference
tol	See reference
maxit	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.374

cyfalg	<i>Fixed-point algorithm for the computation of an M-estimate of multivariate location and scatter</i>
--------	--

Description

See Marazzi A. (1993), p.232

Usage

```
cyfalg(x, a, t, exu = ucv, exv = vcv, exw = wcv, nobs = nrow(x), tau = .dFvGet()$tua,
      maxit = .dFvGet()$mxf, nitmon = .dFvGet()$ntm, iloc = .dFvGet()$ilc,
      icnv = .dFvGet()$icv, tol = .dFvGet()$tlo)
```

Arguments

x	See reference
a	See reference
t	See reference
exu	See reference
exv	See reference
exw	See reference
nobs	See reference
tau	See reference
maxit	See reference
nitmon	See reference
iloc	See reference
icnv	See reference
tol	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.232

cygalg *Conjugate gradient algorithm for the computation of an M-estimate of multivariate location and scatter*

Description

See Marazzi A. (1993), p.238

Usage

```
cygalg(x, a, t, exu = ucv, exup = upcv, exv = vcv, exw = wcv, exwp = wpcv,
      nobs = nrow(x), maxit = .dFvGet()$mxg, nitmon = .dFvGet()$ntm,
      iloc = .dFvGet()$ilc, icnv = .dFvGet()$icv, tol = .dFvGet()$tlo,
      xfud = .dFvGet()$xfd)
```

Arguments

x	See reference
a	See reference
t	See reference
exu	See reference
exup	See reference
exv	See reference
exw	See reference
exwp	See reference
nobs	See reference
maxit	See reference
nitmon	See reference
iloc	See reference
icnv	See reference
tol	See reference
xfud	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.238

cynalg *Newton-type algorithm for the computation of an M-estimate of multivariate location and scatter*

Description

See Marazzi A. (1993), p.235

Usage

```
cynalg(x, a, t, exu = ucv, exup = upcv, exv = vcv, exvp = vpcv,
       exw = wcv, exwp = wpcv, nobs = nrow(x), maxit = .dFvGet()$mxn,
       nitmon = .dFvGet()$ntm, iloc = .dFvGet()$ilc, icnv = .dFvGet()$icv,
       tol = .dFvGet()$tlo, xfud = .dFvGet()$xfd)
```

Arguments

x	See reference
a	See reference
t	See reference
exu	See reference
exup	See reference
exv	See reference
exvp	See reference
exw	See reference
exwp	See reference
nobs	See reference
maxit	See reference
nitmon	See reference
iloc	See reference
icnv	See reference
tol	See reference
xfud	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.235

Dbinom	<i>Diagonal matrix D for the binomial case in discrete GLM</i>
--------	--

Description

See Marazzi A. (1993), p.310

Usage

```
Dbinom(y, ci, vtheta, wa, ni, f0, oi = 0, kap = 1e-06)
```

Arguments

y	See reference
ci	See reference
vtheta	See reference
wa	See reference
ni	See reference
f0	See reference
oi	See reference
kap	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.310

dfcomn	<i>Assigns values to the ROBETH parameters included in common blocks</i>
--------	--

Description

See Marazzi A. (1993), p.405

Usage

```
dfcomn(ipsi = -9, c = -1.345, h1 = -1.7, h2 = -3.4, h3 = -8.5,
       xk = -1.548, d = -1.345, beta = -0.5, bet0 = -1, iucv = -1,
       a2 = 0, b2 = -3, chk = -9, ckw = -2, bb = -1, bt = -1,
       cw = -1, em = -1.345, cr = -2, vk = -1, np = -2, nu = -1,
       v7 = -1, iwww = -1)
```

Arguments

ipsi	Option parameter for the choice of ψ . Set $-4 \leq \text{ipsi} \leq 4$
c	Parameter c of the Huber function
h1	Parameter $h1$ of the Hampel function
h2	Parameter $h2$ of the Hampel function
h3	Parameter $h3$ of the Hampel function
xk	Parameter k of the rescaled Tukey biweight
d	See reference
beta	Parameter β to make σ estimate asymptotically unbiased
bet0	Parameter β_0 to make σ estimate asymptotically unbiased
iucv	Option parameter for the choice of $u(s)$, $u'(s)$, $v(s)$, $v'(s)$, $w(s)$ or $w'(s)$
a2	Parameter a^2 of Huber's minimax u-function
b2	Parameter b^2 of Huber's minimax u-function
chk	Parameter c of the Hampel-Krasker u-function
ckw	Parameter c of the Krasker-Welsch u-function
bb	Parameter b of the Mallows-unstandard u-function
bt	Option parameter for $w(s)$ or $w'(s)$
cw	Option parameter for $w(s)$ or $w'(s)$
em	Parameter em for unstandard u-function
cr	Parameter cr for unstandard u-function
vk	Parameter vk for unstandard u-function
np	Parameter np for unstandard u-function
nu	Parameter nu for unstandard u-function
v7	Parameter v for unstandard u-function
iwww	Option parameter for the choice of $\bar{\omega}$. Set $0 \leq \text{iwww} \leq 3$

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.405

dfrpar	<i>Sets default parameters for regression estimates</i>
--------	---

Description

See Marazzi A. (1993), p.398 and p.406

Usage

```
dfrpar(x, etype, upar = -1, psipar = -1)
```

Arguments

x	See reference
etype	See reference
upar	See reference
psipar	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.398 and p.406

dfvals	<i>Provide default values for most scalar parameters used by the Robeth subroutines</i>
--------	---

Description

See Marazzi A. (1993), p.404

Usage

```
dfvals()
```

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.404

dotp *Forms the scalar (dot) product of two vectors*

Description

See Marazzi A. (1993), p.350

Usage

dotp(x, y, n = nrow(x), incx = 1, incy = 1)

Arguments

x	See reference
y	See reference
n	See reference
incx	See reference
incy	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.350

dotpd *Forms the scalar (dot) product of two vectors (double precision)*

Description

See Marazzi A. (1993), p.350

Usage

dotpd(x, y, n = nrow(x), incx = 1, incy = 1)

Arguments

x	See reference
y	See reference
n	See reference
incx	See reference
incy	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.350

dpoiss

Diagonal matrix D for the Poisson case in discrete GLM

Description

See Marazzi A. (1993), p.312

Usage

```
dpoiss(y, ci, vtheta, wa, f0, oi = 0, kap = 1e-06)
```

Arguments

y	See reference
ci	See reference
vtheta	See reference
wa	See reference
f0	See reference
oi	See reference
kap	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.31

exch	<i>Exchanges two columns of a symmetric matrix</i>
------	--

Description

See Marazzi A. (1993), p.364

Usage

exch(s, n, h, k)

Arguments

s	See reference
n	See reference
h	See reference
k	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.364

exchd	<i>Exchanges two columns of a symmetric matrix (double precision)</i>
-------	---

Description

See Marazzi A. (1993), p.364

Usage

exchd(s, n, h, k)

Arguments

s	See reference
n	See reference
h	See reference
k	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.364

fcum

Cumulative F-distribution function

Description

See Marazzi A. (1993), p.379

Usage

fcum(n1, n2, x)

Arguments

n1	See reference
n2	See reference
x	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.37

Fn.Exp.f *Parametric estimate of survival cdf*

Description

Parametric estimate of survival cdf

Usage

Fn.Exp.f(z, y , delta, mu, sigma, lambda, zero=1e-4)

Arguments

z	See reference
y	See reference
delta	See reference
mu	See reference
sigma	See reference
lambda	See reference
zero	See reference

Value

See reference

References

Marazzi A. (2010) Robust estimation of the extended log-gamma (not yet published)

fstord *Determination of the j-th order statistic*

Description

See Marazzi A. (1993), p.389

Usage

fstord(y, j)

Arguments

y	See reference
j	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.389

gauss	<i>Cumulative Gaussian distribution function</i>
-------	--

Description

See Marazzi A. (1993), p.371

Usage

```
gauss(kode = 1, x)
```

Arguments

kode See reference

x See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.371

gaussd	<i>Cumulative Gaussian distribution function (double precision)</i>
--------	---

Description

See Marazzi A. (1993), p.371

Usage

```
gaussd(kode = 1, x)
```

Arguments

kode	See reference
x	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.371

gfedca	<i>Diagonal matrices D_G and E_G</i>
--------	--------------------------------------

Description

See Marazzi A. (1993), p.309

Usage

```
gfedca(vtheta, ci, wa, ni, oi = 0, ics = .dFvGet()$ics)
```

Arguments

vtheta	See reference
ci	See reference
wa	See reference
ni	See reference
oi	See reference
ics	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.309

gintac *Initial values of theta, A and c_i,...,c_n*

Description

See Marazzi A. (1993), p.292

Usage

```
gintac(x, y, ni, oi = 0, icode = .dFvGet()$ics, maxtt = .dFvGet()$mxt,  
       maxta = .dFvGet()$mxf, tolt = .dFvGet()$tlo, tola = .dFvGet()$tlo,  
       b = 1.1 * sqrt(np), c = 1.345)
```

Arguments

x	See reference
y	See reference
ni	See reference
oi	See reference
icode	See reference
maxtt	See reference
maxta	See reference
toltt	See reference
tolta	See reference
b	See reference
c	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.292

 glmdev

The total deviance of the fitted generalized linear model

Description

Computes the sum of the vector deviance and other intermediate results

Usage

```
glmdev(y, ni, ci, wa, vtheta, offset = 0, ics = .dFvGet()$ics)
```

Arguments

y	The vector of observations
ni	The number of trial at xi in the binomial case (ics=2). Otherwise ni=1 for each xi.
ci	The constants ci
wa	The vector of $a_i = b / \lambda_{xi}$
vtheta	The vector of ξ^T
offset	Optional offset added to the linear predictor.
ics	Set ics=1 for Bernoulli case, ics=2 for Binomial case and ics=3 for Poisson case

Value

A list with the following components:

dev	$2 * \sum_j \text{abs}(L_i - T_i)$
thetas	The estimates of $\theta_{i,j}$
li	The values of L_i
sc	The values of T_i

References

Kuensch, H.R., Stefanski L.A., Carroll R.J. (1989). Conditionally unbiased bounded-influence estimation in general regression models, with application to generalized linear models. *Journal of the American Statistical Association*, 84, 460-466.

Marazzi, A. (1993). *Algorithms, Routines, and S-functions for robust Statistics*. Chapman and Hall, New York.

Marazzi A. (1997). Object oriented S-plus functions for robust discrete generalized linear models available in the doc folder of this package.

gyastp

Fixed-point algorithm for the A-step

Description

See Marazzi A. (1993), p.301

Usage

```
gyastp(x, y, ni, vtheta, ci, a, oi = 0, b = 1.1 * sqrt(nvar),
      iugl = .dFvGet()$iug, ics = .dFvGet()$ics, tau = .dFvGet()$tua,
      maxit = .dFvGet()$mxf, nitmon = .dFvGet()$ntm, icnv = .dFvGet()$icv,
      tol = .dFvGet()$tlo)
```

Arguments

x	See reference
y	See reference
ni	See reference
vtheta	See reference
ci	See reference
a	See reference
oi	See reference
b	See reference
iugl	See reference
ics	See reference
tau	See reference
maxit	See reference
nitmon	See reference
icnv	See reference
tol	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.301

gycstp

Newton-type algorithm for the c-step

Description

See Marazzi A. (1993), p.299

Usage

```
gycstp(icas = .dFvGet()$ics, ialg = .dFvGet()$ilg, ni, a, e,  
       tol = .dFvGet()$tlo, maxit = .dFvGet()$mxt, t)
```

Arguments

icas	See reference
ialg	See reference
ni	See reference
a	See reference
e	See reference
tol	See reference
maxit	See reference
t	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.299

gymain*Main algorithm*

Description

See Marazzi A. (1993), p.304

Usage

```
gymain(x, y, ni, cov, a, theta, oi = 0, b = 1.1 * sqrt(np),
      gam = .dFvGet()$gma, tau = .dFvGet()$tua, icense = .dFvGet()$ics,
      iugl = .dFvGet()$iug, iopt = .dFvGet()$ipo, ialg = .dFvGet()$ilg,
      icnvt = .dFvGet()$icn, icnva = .dFvGet()$icv, maxit = .dFvGet()$mxx,
      maxtt = .dFvGet()$mxt, maxta = .dFvGet()$mxf, maxtc = .dFvGet()$mxt,
      nitmnt = .dFvGet()$ntm, nitmna = .dFvGet()$ntm, tol = .dFvGet()$tlo,
      tolt = .dFvGet()$tlo * 10, tola = .dFvGet()$tlo * 10,
      tolc = .dFvGet()$tlo * 10)
```

Arguments

x	See reference
y	See reference
ni	See reference
cov	See reference
a	See reference
theta	See reference
oi	See reference
b	See reference
gam	See reference
tau	See reference
icense	See reference
iugl	See reference
iopt	See reference
ialg	See reference
icnvt	See reference
icnva	See reference
maxit	See reference
maxtt	See reference
maxta	See reference
maxtc	See reference
nitmnt	See reference
nitmna	See reference
tol	See reference
tolc	See reference
tolt	See reference
tola	See reference
tolc	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.304

gytstp *Newton-type algorithm for the theta-step*

Description

See Marazzi A. (1993), p.295

Usage

```
gytstp(x, y, ci, theta, wa, cov, ni, oi = 0, gam = .dFvGet()$gma,
      tol = .dFvGet()$tlo, tau = .dFvGet()$tua, iopt = .dFvGet()$ipo,
      icase = .dFvGet()$ics, icnv = .dFvGet()$icn, maxit = .dFvGet()$mxt,
      nitmon = .dFvGet()$ntm)
```

Arguments

x	See reference
y	See reference
ci	See reference
theta	See reference
wa	See reference
cov	See reference
ni	See reference
oi	See reference
gam	See reference
tol	See reference
tau	See reference
iopt	See reference
icase	See reference
icnv	See reference
maxit	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.295

h12 *Constructs and/or applies a single elementary Householder transformation*

Description

See Marazzi A. (1993), p.359

Usage

h12(mode, lpivot, l1, u, up, c, ice, icv, ncv)

Arguments

mode	See reference
lpivot	See reference
l1	See reference
u	See reference
up	See reference
c	See reference
ice	See reference
icv	See reference
ncv	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.359

h12d *Constructs and/or applies a single elementary Householder transformation (double precision)*

Description

See Marazzi A. (1993), p.359

Usage

h12d(mode, lpivot, l1, u, up, c, ice, icv, ncv)

Arguments

mode	See reference
lpivot	See reference
l1	See reference
u	See reference
up	See reference
c	See reference
ice	See reference
icv	See reference
ncv	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.35

 hylmse

Resampling algorithm for the computation of the LMS estimate

Description

See Marazzi A. (1993), p.208

Usage

```
hylmse(x, y, nq = np, ik = .dFvGet()$ik1, iopt = .dFvGet()$ipt,
      intch = .dFvGet()$ich, nrep, tol = .dFvGet()$tlo, tau = .dFvGet()$tua,
      iseed = .dFvGet()$isd)
```

Arguments

x	See reference
y	See reference
nq	See reference
ik	See reference
iopt	See reference
intch	See reference
nrep	See reference
tol	See reference
tau	See reference
iseed	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.208

 hylvse

Resampling algorithm for the computation of the LTS estimate

Description

See Marazzi A. (1993), p.212

Usage

```
hylvse(x, y, nq = np, ik = .dFvGet()$ik1, iopt = .dFvGet()$ipt,
      intch = .dFvGet()$ich, nrep, tol = .dFvGet()$tlo, tau = .dFvGet()$tua,
      iseed = .dFvGet()$isd, smin)
```

Arguments

x	See reference
y	See reference
nq	See reference
ik	See reference
iopt	See reference
intch	See reference
nrep	See reference
tol	See reference
tau	See reference
iseed	See reference
smin	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.212

hysest

Resampling algorithm for the computation of S-estimates

Description

See Marazzi A. (1993), p.216

Usage

```
hysest(x, y, nq = np, iopt = .dFvGet()$ipt, intch = .dFvGet()$ich,
      nrep, tols = .dFvGet()$tls, tolr = .dFvGet()$tlr, tau = .dFvGet()$tua,
      gam = .dFvGet()$gma, maxit = .dFvGet()$mxt, maxs1 = .dFvGet()$msx,
      maxs2 = .dFvGet()$mxx, expsi = psi, expsp = psp, exchi = chi,
      iseed = .dFvGet()$isd)
```

Arguments

x	See reference
y	See reference
nq	See reference
iopt	See reference
intch	See reference
nrep	See reference
tols	See reference
tolr	See reference
tau	See reference
gam	See reference
maxit	See reference
maxs1	See reference
maxs2	See reference
expsi	See reference
exsp	See reference
exchi	See reference
iseed	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.216

hysestw *Resampling algorithm for the computation of weighted S-estimates*

Description

See Marazzi A. (1993), p.216

Usage

```
hysestw(x, y, wgt, nq = np, iopt = .dFvGet()$ipt, intch = .dFvGet()$ich,
        nrep, tols = .dFvGet()$tls, tolr = .dFvGet()$tlr, tau = .dFvGet()$tua,
        gam = .dFvGet()$gma, maxit = .dFvGet()$mxt, maxs1 = .dFvGet()$msx,
        maxs2 = .dFvGet()$mxs, expsi = psi, expsp = psp, exchi = chi,
        iseed = .dFvGet()$isd)
```

Arguments

x	See reference
y	See reference
wgt	See reference
nq	See reference
iopt	See reference
intch	See reference
nrep	See reference
tols	See reference
tolr	See reference
tau	See reference
gam	See reference
maxit	See reference
maxs1	See reference
maxs2	See reference
expsi	See reference
exsp	See reference
exchi	See reference
iseed	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.216

ingama	<i>Incomplete Gamma-integral function</i>
--------	---

Description

See Marazzi A. (1993), p.381

Usage

ingama(x, p)

Arguments

x	See reference
p	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.381

kfascv	<i>Backtransformation of the covariance matrix of the coefficient estimates</i>
--------	---

Description

See Marazzi A. (1993), p.152

Usage

kfascv(xt, cov, k = np, mdx = nrow(xt), f = .dFvGet())\$fff, sg, ip)

Arguments

xt	See reference
cov	See reference
k	See reference
mdx	See reference
f	See reference
sg	See reference
ip	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.152

kfedcb

Diagonal hat matrices D_M , E_M , D_S , and E_S

Description

See Marazzi A. (1993), p.159

Usage

```
kfedcb(wgt, rs, expsi = psi, expsp = psp, sigma, itype = .dFvGet()$ite)
```

Arguments

wgt	See reference
rs	See reference
expsi	See reference
expsp	See reference
sigma	See reference
itype	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.15

kfedcc

Diagonal 'check' matrices D_M, E_M, D_S, and E_S

Description

See Marazzi A. (1993), p.160

Usage

```
kfedcc(wgt, rs, expsi = psi, expsp = psp, sigma, itype = .dFvGet()$ite)
```

Arguments

wgt	See reference
rs	See reference
expsi	See reference
expsp	See reference
sigma	See reference
itype	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.160

kffacv

Correction factor f_H for the covariance matrix of a Huber-type estimate

Description

See Marazzi A. (1993), p.154

Usage

```
kffacv(rs, expsi = psi, expsp = psp, np, sigma)
```

Arguments

rs	See reference
expsi	See reference
expsp	See reference
np	See reference
sigma	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.154

kiascv	<i>Covariance matrix of the coefficient estimates of the form $f.inv(XT X)$ in the transformed coordinate system</i>
--------	---

Description

See Marazzi A. (1993), p.150

Usage

```
kiascv(xt, k = np, mdx = nrow(xt), fu = .dFvGet()$fu1, fb = .dFvGet()$fb1)
```

Arguments

xt	See reference
k	See reference
mdx	See reference
fu	See reference
fb	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.150

kiedch	<i>Diagonal matrices D_M, E_M, D_S, E_S when ψ is the Huber function</i>
--------	---

Description

See Marazzi A. (1993), p.156

Usage

```
kiedch(wgt, c = .dFvGet()$ccc, itype = .dFvGet()$ite)
```

Arguments

wgt See reference

c See reference

itype See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.156

kiedcu	<i>Diagonal matrices D_M, E_M, D_S, E_S when ψ is a user-supplied function</i>
--------	---

Description

See Marazzi A. (1993), p.157

Usage

```
kiedcu(wgt, expsi = psi, itype = .dFvGet()$ite, upper = .dFvGet()$upr,
      til = .dFvGet()$tli)
```

Arguments

wgt See reference

expsi See reference

itype See reference

upper See reference

til See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.157

ktaskv	<i>Covariance matrix of the coefficient estimates of the form $f.inv(XT X)$</i>
--------	--

Description

See Marazzi A. (1993), p.147

Usage

```
ktaskv(x, n = nrow(x), tau = .dFvGet()$tua, f = .dFvGet()$fff)
```

Arguments

x	See reference
n	See reference
tau	See reference
f	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.147

ktaskw	<i>Covariance matrix of the coefficient estimates of the form $f \cdot \text{inv}(S1) S2 \text{inv}(S1)$</i>
--------	---

Description

See Marazzi A. (1993), p.148

Usage

```
ktaskw(x, d, e, tau = .dFvGet()$tua, ia = .dFvGet()$ia1, f = .dFvGet()$fff,  
      f1 = .dFvGet()$ff1, iainv = .dFvGet()$ia2, a)
```

Arguments

x	See reference
d	See reference
e	See reference
tau	See reference
ia	See reference
f	See reference
f1	See reference
iainv	See reference
a	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.148

lgama	<i>Logarithm at the Gamma-function at the point x</i>
-------	---

Description

See Marazzi A. (1993), p.383

Usage

lgama(x)

Arguments

x See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.383

libet0	<i>Computation of Beta0 = Phi_inv(0.75)</i>
--------	---

Description

See Marazzi A. (1993), p.46

Usage

libet0()

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.46

libeth	<i>Computation of Int Chi(s) dPhi(s) when Chi=Psi.Psi/2 and Psi is the Huber function</i>
--------	---

Description

See Marazzi A. (1993), p.44

Usage

```
libeth(d = .dFvGet())$ddd)
```

Arguments

d See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.44

libetu	<i>Computation of Int Chi(s) dPhi(s) when Chi is a user-supplied function</i>
--------	---

Description

See Marazzi A. (1993), p.45

Usage

```
libetu(exchi = chi, upper = .dFvGet())$upr, til = .dFvGet())$tli)
```

Arguments

exchi See reference
upper See reference
til See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.45

 liclls

Classical estimates of mean and standard deviation

Description

See Marazzi A. (1993), p.27

Usage

liclls(y)

Arguments

y Vector of observations

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.27

 liepsh

Computation of $\int \Psi(s) \cdot \Psi(s) \, d\Phi(s)$ and $\int \Psi'(s) \, d\Phi(s)$ when Ψ is the Huber function

Description

See Marazzi A. (1993), p.47

Usage

liepsh(c = .dFvGet())\$ccc)

Arguments

c See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.47

liepsu	<i>Computation of $\int \Psi(s) \cdot \Psi(s) d\Phi(s)$ and $\int \Psi'(s) d\Phi(s)$ when Ψ is a user-supplied external function</i>
--------	--

Description

See Marazzi A. (1993), p.48

Usage

```
liepsu(expsi = psi, upper = .dFvGet()$upr, til = .dFvGet()$tli)
```

Arguments

expsi	See reference
upper	See reference
til	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.48

liindh	<i>Inverts the approximate null distribution of the one-sample Wilcoxon test statistic</i>
--------	--

Description

See Marazzi A. (1993), p.36

Usage

```
liindh(alpha = .dFvGet()$alf, n)
```

Arguments

alpha See reference
n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.36

liinds *Inverts the approximate null distribution of the sign test statistic*

Description

See Marazzi A. (1993), p.35

Usage

```
liinds(alpha = .dFvGet())$alf, n)
```

Arguments

alpha See reference
n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.35

liindw	<i>Inverts the approximate null distribution of the Mann-Whitney test statistic</i>
--------	---

Description

See Marazzi A. (1993), p.43

Usage

```
liindw(alpha = .dFvGet())$alf, m, n)
```

Arguments

alpha	See reference
m	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.43

lilars	<i>Median an median absolute deviation</i>
--------	--

Description

See Marazzi A. (1993), p.28

Usage

```
lilars(y, isort = .dFvGet())$isr)
```

Arguments

y	See reference
isort	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.28

littst	<i>t-test for the shift parameter</i>
--------	---------------------------------------

Description

See Marazzi A. (1993), p.37

Usage

```
littst(x, y, alpha = .dFvGet())$alf)
```

Arguments

x	See reference
y	See reference
alpha	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.37

lmdd	<i>Median and median absolute deviation</i>
------	---

Description

See Marazzi A. (1993), p.388

Usage

```
lmdd(x, isort = 1)
```

Arguments

x	See reference
isort	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.388

Irfctd	<i>Computation of Li, li and lip</i>
--------	--------------------------------------

Description

See Marazzi A. (1993), p.282-286 and p.297-298

Usage

Irfctd(icase, y, ci, vtheta, offset, wa, ni, i0, i1, i2)

Arguments

icase	Integer: 1 for Bernouilli, 2 for binomial and 3 for Poisson.
y	The y vector.
ci	The c_i vector.
vtheta	The x by theta vector.
offset	The offset vector.
wa	The a_i vector.
ni	The integer n_i vector.
i0	Integer: 1 to compute Li otherwise 0.
i1	Integer: 1 to compute li otherwise 0.
i2	Integer: 1 to compute lip otherwise 0.

Value

List with the following components :

f0	NULL if i0=0 else Li.
i1	NULL if i1=0 else li , derivative of Li.
i2	NULL if i2=0 else lip, derivative of li.
sf0	NULL if i0=0 else sum of the Li components.

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.282-286 and p.297-298

lyhalg

M-estimate of location with simultaneous estimation of scale

Description

See Marazzi A. (1993), p.30

Usage

```
lyhalg(y, expsi = psi, expsp = psp, exchi = chi, theta, sigmai,  
      tol = .dFvGet()$tlo, gam = .dFvGet()$gma, isigma = .dFvGet()$isg,  
      maxit = .dFvGet()$mxt, maxis = .dFvGet()$mxs)
```

Arguments

y	See reference
expsi	See reference
exsp	See reference
exchi	See reference
theta	See reference
sigmai	See reference
tol	See reference
gam	See reference
isigma	See reference
maxit	See reference
maxis	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.30

lyhdle	<i>Hodges-Lehman estimate and confidence intervals for the center of symmetry based on the one-sample Wilcoxon test</i>
--------	---

Description

See Marazzi A. (1993), p.33

Usage

```
lyhdle(y, isort = .dFvGet()$isr, k, tol = .dFvGet()$tlo,
       maxit = .dFvGet()$mxt)
```

Arguments

y	See reference
isort	See reference
k	See reference
tol	See reference
maxit	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.33

lymnwt	<i>Nonparametric estimate and confidence intervals for the shift parameter based on the Mann-Whitney test statistic</i>
--------	---

Description

See Marazzi A. (1993), p.41

Usage

```
lymnwt(x, y, isort = .dFvGet()$isr, k, tol = .dFvGet()$tlo,
       maxit = .dFvGet()$mxt)
```


Arguments

x	See reference
y	See reference
isort	See reference
k	See reference
tol	See reference
maxit	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.41

lytau2	<i>tau-test for the shift parameter</i>
--------	---

Description

See Marazzi A. (1993), p.38

Usage

```
lytau2(z, expsi = psi, expsp = psp, exchi = chi, exrho = rho, m, n,
      tol = .dFvGet()$tlo, gam = .dFvGet()$gma, isigma = .dFvGet()$isg,
      maxit = .dFvGet()$mxt, nitmon = .dFvGet()$ntm)
```

Arguments

z	See reference
expsi	See reference
expsp	See reference
exchi	See reference
exrho	See reference
m	See reference
n	See reference
tol	See reference
gam	See reference
isigma	See reference
maxit	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.38

 lywalg

W-algorithm for M-estimate of location

Description

Robust location estimate with simultaneous estimation of the scale parameter

Usage

```
lywalg(y, lambda, psp0 = psp(0), expsi = psi, exchi = chi, exrho = rho,
       sigmai, tol = .dFvGet()$tlo, gam = .dFvGet()$gma,
       isigma = .dFvGet()$isg, maxit = .dFvGet()$mxt, maxis = .dFvGet()$mxs,
       nitmon = .dFvGet()$ntm)
```

Arguments

y	Vector containing the observations
lambda	Initial solution of the location parameter
psp0	Value of psp(0) (first derivative of the psi function)
expsi	User supplied psi function
exchi	User supplied chi function
exrho	User supplied rho function
sigmai	Initial estimate of the scale parameter
tol	Relative precision for the convergence criterion
gam	Relaxation factor. Set $0 < \text{gam} < 2$.
isigma	If isigma<0, the value of sigma is not changed during the first iteration. If isigma=0, bypass iteration on sigma (sigmaf=sigmai). If lisigma>0, sigma is updated using the robeth function rysigm.
maxit	Maximum number of cycles
maxis	Maximum number of iterations for the scale step
nitmon	If nitmon>0 and the iteration counter is a multiple of nitmon, the current value of sigma, theta and delta are printed. If no iteration monitoring is required, set nitmon equal to 0.

Details

The .dFv variables for the default values must be created by a call to the `dfvals()` function of the `robeth` package. To see if this variable is available in your R session, type `ls(all.names=TRUE)`. The parameters for `psi`, `chi` and `rho` functions must also be set by a preliminary call to the `dfconn` function of the `robeth` package.

Value

<code>lambda</code>	Final value of the location estimate
<code>nit</code>	Reached number of cycles
<code>sigmaf</code>	Final estimate of sigma
<code>rs</code>	The residual vector

References

Marazzi A. (1993), *Algorithm, Routines, and S functions for Robust Statistics*, Wadsworth & Brooks/cole, Pacific Grove, California. p.30 and p.83 .

 mchl

Cholesky decomposition of a symmetric matrix

Description

See Marazzi A. (1993), p.353

Usage

`mchl(a, n)`

Arguments

<code>a</code>	See reference
<code>n</code>	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.353

mchld *Cholesky decomposition of a symmetric matrix (double precision)*

Description

See Marazzi A. (1993), p.353

Usage

mchld(a, n)

Arguments

a See reference
n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.353

messagena *Print a message when a required argument is missing*

Description

Function only needed for the interface

Usage

messagena(x)

Arguments

x A character string

Value

None (invisible NULL).

mff *Multiplies a full matrix by a full matrix*

Description

See Marazzi A. (1993), p.339

Usage

```
mff(a, b, m = nrow(a))
```

Arguments

a	See reference
b	See reference
m	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.339

mffd *Multiplies a full matrix by a full matrix (double precision)*

Description

See Marazzi A. (1993), p.339

Usage

```
mffd(a, b, m = nrow(a))
```

Arguments

a	See reference
b	See reference
m	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.339

mfragr

Generation and comparison of all regressions on subsets of covariates

Description

See Marazzi A. (1993), p.258

Usage

```
mfragr(x, y, vp, nc, itype = .dFvGet()$ith, c = .dFvGet()$ccc,
      tol = .dFvGet()$tlo, gam = .dFvGet()$gma,
      maxit = .dFvGet()$mxt, sigmac, sigmar)
```

Arguments

x	See reference
y	See reference
vp	See reference
nc	See reference
itype	See reference
c	See reference
tol	See reference
gam	See reference
maxit	See reference
sigmac	See reference
sigmar	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.258

mfy *Multiplies a full matrix by a vector*

Description

See Marazzi A. (1993), p.342

Usage

mfy(a, y, m = nrow(a), iye = 1, ize = 1)

Arguments

a	See reference
y	See reference
m	See reference
iye	See reference
ize	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.342

mfyd *Multiplies a full matrix by a vector (double precision)*

Description

See Marazzi A. (1993), p.342

Usage

mfyd(a, y, m = nrow(a), iye = 1, ize = 1)

Arguments

a	See reference
y	See reference
m	See reference
iye	See reference
ize	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.342

mhat

Computes the diagonal elements of the hat matrix

Description

See Marazzi A. (1993), p.354

Usage

mhat(x, n = nrow(x), k = np, sh)

Arguments

x	See reference
n	See reference
k	See reference
sh	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.354

minv *Inverts a triangular matrix*

Description

See Marazzi A. (1993), p.348

Usage

```
minv(r, n, tau = .dFvGet())$tua)
```

Arguments

r	See reference
n	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.348

minvd *Inverts a triangular matrix (double precision)*

Description

See Marazzi A. (1993), p.348

Usage

```
minvd(r, n, tau = .dFvGet())$tua)
```

Arguments

r	See reference
n	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.348

mirtsr

Computation of (robust) t-statistics for t-directed search

Description

See Marazzi A. (1993), p.262

Usage

```
mirtsr(x, y, itype = .dFvGet()$ite, c = .dFvGet()$ccc,
       d = .dFvGet()$ddd, tol = .dFvGet()$tlo,
       gam = .dFvGet()$gma, maxit = .dFvGet()$mxt,
       maxis = .dFvGet()$mxs, tau = .dFvGet()$tua)
```

Arguments

x	See reference
y	See reference
itype	See reference
c	See reference
d	See reference
tol	See reference
gam	See reference
maxit	See reference
maxis	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.262

mly *Multiplies a lower-triangular matrix by a vector*

Description

See Marazzi A. (1993), p.346

Usage

```
mly(a, y, n, iye = 1)
```

Arguments

a	See reference
y	See reference
n	See reference
iye	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.346

mlyd *Multiplies a lower-triangular matrix by a vector (double precision)*

Description

See Marazzi A. (1993), p.346

Usage

```
mlyd(a, y, n, iye = 1)
```

Arguments

a	See reference
y	See reference
n	See reference
iye	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.346

msf

Multiplies a symmetric matrix by a full matrix

Description

See Marazzi A. (1993), p.340

Usage

msf(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.340

msf1 *Multiplies a symmetric matrix by a full matrix when the result is a symmetric matrix*

Description

See Marazzi A. (1993), p.341

Usage

msf1(a, b)

Arguments

a See reference
b See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.341

msf1d *Multiplies a symmetric matrix by a full matrix when the result is a symmetric matrix*

Description

See Marazzi A. (1993), p.341

Usage

msf1d(a, b)

Arguments

a See reference
b See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.341

msfd *Multiplies a symmetric matrix by a full matrix (double precision)*

Description

See Marazzi A. (1993), p.340

Usage

msfd(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California.v

mss *Multiplies a symmetric matrix by a symmetric matrix*

Description

See Marazzi A. (1993), p.338

Usage

mss(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.338

mssd	<i>Multiplies a symmetric matrix by a symmetric matrix (double precision)</i>
------	---

Description

See Marazzi A. (1993), p.342

Usage

mssd(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.342

mtt1

Multiplies an upper-triangular matrix by its transpose

Description

See Marazzi A. (1993), p.343

Usage

mtt1(a, n)

Arguments

a See reference
n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.343

mtt1d

Multiplies an upper-triangular matrix by its transpose (double precision)

Description

See Marazzi A. (1993), p.343

Usage

mtt1d(a, n)

Arguments

a See reference
n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.343

mtt2

Multiplies a lower-triangular matrix by its transpose

Description

See Marazzi A. (1993), p.344

Usage

mtt2(a, n)

Arguments

a See reference

n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.344

mtt2d

Multiplies a lower-triangular matrix by its transpose (double precision)

Description

See Marazzi A. (1993), p.344

Usage

mtt2d(a, n)

Arguments

a See reference

n See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.344

mtt3

Multiplies a triangular matrix by a triangular matrix

Description

See Marazzi A. (1993), p.345

Usage

mtt3(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.345

mtt3d *Multiplies a triangular matrix by a triangular matrix (double precision)*

Description

See Marazzi A. (1993), p.345

Usage

mtt3d(a, b, n)

Arguments

a	See reference
b	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.345

mty *Multiplies an upper-triangular matrix by a vector*

Description

See Marazzi A. (1993), p.347

Usage

mty(a, y, n, iye = 1)

Arguments

a	See reference
y	See reference
n	See reference
iye	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.347

mtyd

Multiplies an upper-triangular matrix by a vector

Description

See Marazzi A. (1993), p.347

Usage

```
mtyd(a, y, n, iye = 1)
```

Arguments

a	See reference
y	See reference
n	See reference
iye	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.347

myhbhe	<i>High breakdown point and high efficiency regression with test for bias</i>
--------	---

Description

See Marazzi A. (1993), p.270

Usage

```
myhbhe(x, y, iseed = .dFvGet())$isd)
```

Arguments

x	See reference
y	See reference
iseed	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.270

mymv1m	<i>Simultaneous computation of the MVE and LMS estimates</i>
--------	--

Description

See Marazzi A. (1993), p.265

Usage

```
mymv1m(x, y, ilms = .dFvGet())$ilm, iopt = .dFvGet())$ipt,  
  intch = .dFvGet())$ich, nrep, tolv = .dFvGet())$tlv,  
  tolm = .dFvGet())$tlm, tau = .dFvGet())$tua,  
  iseed = .dFvGet())$isd)
```

Arguments

x	See reference
y	See reference
ilms	See reference
iopt	See reference
intch	See reference
nrep	See reference
tolv	See reference
tolm	See reference
tau	See reference
iseed	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.265

nlgm

Logarithm of the Gamma-function at the point n/2

Description

See Marazzi A. (1993), p.382

Usage

nlgm(n)

Arguments

n	See reference
---	---------------

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.382

nrm2	<i>Forms the Euclidean norm of a vector</i>
------	---

Description

See Marazzi A. (1993), p.351

Usage

```
nrm2(x, n = nrow(x), incx = 1)
```

Arguments

x	See reference
n	See reference
incx	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.351

nrm2d	<i>Forms the Euclidean norm of a vector (double precision)</i>
-------	--

Description

See Marazzi A. (1993), p.351

Usage

```
nrm2d(x, n = nrow(x), incx = 1)
```

Arguments

x	See reference
n	See reference
incx	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.351

permc

Permutes the columns of a matrix by means of transpositions

Description

See Marazzi A. (1993), p.365

Usage

```
permc(x, it, n = nrow(x), iopt = 1)
```

Arguments

x	See reference
it	See reference
n	See reference
iopt	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.365

permv

Permutes the elements of a vector

Description

See Marazzi A. (1993), p.366

Usage

```
permv(y, it, iopt = 1)
```


Arguments

y	See reference
it	See reference
iopt	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.366

poissn	<i>Poisson distribution</i>
--------	-----------------------------

Description

See Marazzi A. (1993), p.368

Usage

```
poissn(lambda, k)
```

Arguments

lambda	See reference
k	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.368

precd	<i>Algorithmic determination of the smallest double precision positive number x</i>
-------	--

Description

See Marazzi A. (1993), p.385

Usage

precd()

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.385

prec	<i>Algorithmic determination of the smallest double precision positive number x</i>
------	--

Description

See Marazzi A. (1993), p.385

Usage

prec()

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.385

probst	<i>Cumulative t-distribution function</i>
--------	---

Description

See Marazzi A. (1993), p.377

Usage

probst(x, ifn)

Arguments

x	See reference
ifn	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.377

Psi	<i>psi weight function for location and regression</i>
-----	--

Description

See Marazzi A. (1993), p.319

Usage

Psi(svls)

Arguments

svls	A vector of input values
------	--------------------------

Value

The values of the psi weight function for each element of svls

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.319

psi *psi weight function for location and regression*

Description

See Marazzi A. (1993), p.319

Usage

psi(s)

Arguments

s A scalar input value

Value

The value of the psi weight function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.319

Psp *psi' weight function for location and regression*

Description

See Marazzi A. (1993), p.320

Usage

Psp(svals)

Arguments

svals A vector of input values

Value

The values of the psi' weight function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.320

psp *psi' weight function for location and regression*

Description

See Marazzi A. (1993), p.320

Usage

psp(s)

Arguments

s	A scalar input value
---	----------------------

Value

The value of the psi' weight function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.320

QD2coef.f *Auxiliary function to find mu and sigma parameter for extended loggamma distribution*

Description

QD2coef.f computes for a given lambda, the mu and sigma parameters

Usage

QD2coef.f(lambda, yc, delta, muI, sigmaI, zero=1e-4)

Arguments

lambda	See reference
yc	See reference
delta	See reference
muI	See reference
sigmaI	See reference
zero	See reference

Value

See reference

References

Marazzi A. (2010) Robust estimation of the extended log-gamma (not yet published)

QD2funC.f	<i>Auxiliary function to find lambda parameter for extended loggamma distribution</i>
-----------	---

Description

QD2funC.f computes a sum of squared residuals for a given lambda

Usage

```
QD2funC.f(lambda, yc, delta, muI, sigmaI, zero=1e-4)
```

Arguments

lambda	See reference
yc	See reference
delta	See reference
muI	See reference
sigmaI	See reference
zero	See reference

Value

See reference

References

Marazzi A. (2010) Robust estimation of the extended log-gamma (not yet published)

Qn.Exp.f

Auxiliary function to compute quantiles of survival cdf

Description

Qn.Exp.f computes quantiles of survival cdf

Usage

Qn.Exp.f(p, yc, delta, mu, sigma, lambda, zero=1e-4)

Arguments

p	See reference
yc	See reference
delta	See reference
mu	See reference
sigma	See reference
lambda	See reference
zero	See reference

Value

See reference

References

Marazzi A. (2010) Robust estimation of the extended log-gamma (not yet published)

quant

Inverse of the standard Gaussian cumulative distribution function

Description

See Marazzi A. (1993), p.372

Usage

quant(p)

Arguments

p	See reference
---	---------------

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.372

Random

Uniform random number generator

Description

See Marazzi A. (1993), p.386

Usage

```
Random(iseed = .dFvGet())$isd)
```

Arguments

iseed See reference

Details

See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California.p.386

`Regtau.f`*Auxiliary function for the computation of QQopt*

Description

QQopt is a resampling algorithm for the determination of the parameters of a Loggamma model

Usage

```
Regtau.f(x, y, b1, c1, b2, c2, N, tol = 1e-6, seed = 567)
```

Arguments

x	See reference
y	See reference
b1	See reference
c1	See reference
b2	See reference
c2	See reference
N	See reference
tol	See reference
seed	See reference

Value

See reference

References

Marazzi A. (2009) Robust estimation of the generalized log-gamma (not yet published)

`RegtauW.f`*Auxiliary function for the computation of QQopt*

Description

QQopt is a resampling algorithm for the determination of the parameters of a Loggamma model

Usage

```
RegtauW.f(x, y, w, b1, c1, b2, c2, N, tol = 1e-6, seed = 567)
```

Arguments

x	See reference
y	See reference
w	See reference
b1	See reference
c1	See reference
b2	See reference
c2	See reference
N	See reference
tol	See reference
seed	See reference

Value

See reference

References

Marazzi A. (2009) Robust estimation of the generalized log-gamma (not yet published)

Rho	<i>Rho weight function for location and regression</i>
-----	--

Description

See Marazzi A. (1993), p.320

Usage

Rho(svals)

Arguments

svals A vector of input values

Value

The values of the rho function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.320

rho	<i>rho weight function for location and regression</i>
-----	--

Description

See Marazzi A. (1993), p.320

Usage

rho(s)

Arguments

s A scalar input value

Value

The value of the rho function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.320

ribet0	<i>Computation of the constant Beta0</i>
--------	--

Description

See Marazzi A. (1993), p.100

Usage

```
ribet0(wgt, itype = .dFvGet()$ite, isqw = .dFvGet()$isq,
      tol = .dFvGet()$tlo)
```

Arguments

wgt See reference
itype See reference
isqw See reference
tol See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.10

ribeth	<i>Computation of the constant Beta when $Chi=Psi.Psi/2$ and Psi is the Huber function</i>
--------	---

Description

See Marazzi A. (1993), p.97

Usage

```
ribeth(wgt, d = .dFvGet()$ddd, itype = .dFvGet()$ite)
```

Arguments

wgt	See reference
d	See reference
itype	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.97

ribetu	<i>Computation of the constant Beta when Chi is a user-supplied function</i>
--------	--

Description

See Marazzi A. (1993), p.98

Usage

```
ribetu(wgt, exchi = chi, itype = .dFvGet()$ite, upper = .dFvGet()$upr,
      til = .dFvGet()$tli)
```

Arguments

wgt	See reference
exchi	See reference
itype	See reference
upper	See reference
til	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.98

riclls	<i>Solution of the least squares problem</i>
--------	--

Description

See Marazzi A. (1993), p.67

Usage

```
riclls(xt, y, k = np, ix = .dFvGet()$ix1, iy = .dFvGet()$iy1,
      sf, sg, sh, ip)
```

Arguments

xt	See reference
y	See reference
k	See reference
ix	See reference
iy	See reference
sf	See reference
sg	See reference
sh	See reference
ip	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.67

 rilars

Solution of the least absolute residual problem

Description

See Marazzi A. (1993), p.71

Usage

```
rilars(x, y, tol = .dFvGet()$tlu)
```

Arguments

x	See reference
y	See reference
tol	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.71

 rimtrd

Double precision version of RIMTRF

Description

See Marazzi A. (1993), p.64

Usage

```
rimtrd(x, n = nrow(x), intch = .dFvGet()$ith, tau = .dFvGet()$tua)
```

Arguments

x	See reference
n	See reference
intch	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.64

See Also

[rimtrf](#)

rimtrf	<i>Upper triangularization (QR-decomposition) of the design matrix and determination of its pseudorank</i>
--------	--

Description

See Marazzi A. (1993), p.64

Usage

```
rimtrf(x, n = nrow(x), intch = .dFvGet()$ith, tau = .dFvGet()$tua)
```

Arguments

x	See reference
n	See reference
intch	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.64

rmvc	<i>Removes a column from a transformed design matrix and updates its QR-decomposition</i>
------	---

Description

See Marazzi A. (1993), p.357

Usage

```
rmvc(x, n = nrow(x), l, j, ip)
```

Arguments

x	See reference
n	See reference
l	See reference
j	See reference
ip	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.357

ruben	<i>Cumulative distribution and density function of a linear combination of chi-2 random variables</i>
-------	---

Description

See Marazzi A. (1993), p.375

Usage

```
ruben(xlmbda, delta, mult, x, xmode = 1, maxit = 50, eps = 1e-04)
```


Arguments

xlmbda	See reference
delta	See reference
mult	See reference
x	See reference
xmode	See reference
maxit	See reference
eps	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.375

rybifr	<i>Bounded influence regression</i>
--------	-------------------------------------

Description

See Marazzi A. (1993), p.410

Usage

```
rybifr(x, y, np, nthet = np + 1, itype = 2, icoll = 0, isigma = 1,
      ch = 1.345, ck = 1.05 * sqrt(nthet), bm = 1.05 * sqrt(nthet),
      tol = 0.001, tau = 1e-06, maxitt = 50, maxitw = 80)
```

Arguments

x	See reference
y	See reference
np	See reference
nthet	See reference
itype	See reference
icoll	See reference
isigma	See reference
ch	See reference
ck	See reference

bm	See reference
tol	See reference
tau	See reference
maxitt	See reference
maxitw	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.410

ryhalg	<i>H-algorithm for M-estimates</i>
--------	------------------------------------

Description

See Marazzi A. (1993), p.98

Usage

```
ryhalg(x, y, theta, wgt, cov, expsi = psi, exchi = chi, exrho = rho,
       sigmai, k = np, tol = .dFvGet()$tlo, gam = .dFvGet()$gma,
       tau = .dFvGet()$tua, itype = .dFvGet()$ite, ix = .dFvGet()$ix1,
       iy = .dFvGet()$iy1, ic = .dFvGet()$ic1, isigma = .dFvGet()$isg,
       icnv = .dFvGet()$icn, maxit = .dFvGet()$mxt, maxis = .dFvGet()$mxs,
       nitmon = .dFvGet()$ntm, sf, sg, sh, ip)
```

Arguments

x	See reference
y	See reference
theta	See reference
wgt	See reference
cov	See reference
expsi	See reference
exchi	See reference
exrho	See reference
sigmai	See reference
k	See reference

tol	See reference
gam	See reference
tau	See reference
itype	See reference
ix	See reference
iy	See reference
ic	See reference
isigma	See reference
icnv	See reference
maxit	See reference
maxis	See reference
nitmon	See reference
sf	See reference
sg	See reference
sh	See reference
ip	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.98

 ryalg

Newton algorithm with adaptive steps for M-estimates

Description

See Marazzi A. (1993), p.73

Usage

```
rynalg(x, y, theta, wgt, cov, expsi = psi, expsp = psp, exchi = chi,
      exrho = rho, sigmai, gam = .dFvGet()$gma, tol = .dFvGet()$tlo,
      tau = .dFvGet()$tua, itype = .dFvGet()$ite, iopt = .dFvGet()$iop,
      isigma = .dFvGet()$isg, icnv = .dFvGet()$icn, maxit = .dFvGet()$mxt,
      maxis = .dFvGet()$mxs, nitmon = .dFvGet()$ntm)
```

Arguments

x	See reference
y	See reference
theta	See reference
wgt	See reference
cov	See reference
expsi	See reference
exsp	See reference
exchi	See reference
exrho	See reference
sigmai	See reference
gam	See reference
tol	See reference
tau	See reference
itype	See reference
iopt	See reference
isigma	See reference
icnv	See reference
maxit	See reference
maxis	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.73

rysalg

S-algorithm for M-estimates

Description

See Marazzi A. (1993), p.87

Usage

```
rysalg(x, y, theta, wgt, cov, sigmai, tol = .dFvGet()$tlo,
      tau = .dFvGet()$tua, itype = .dFvGet()$ite,
      isigma = .dFvGet()$isg, icnv = .dFvGet()$icn,
      maxit = .dFvGet()$mxt, maxis = .dFvGet()$mxs,
      nitmon = .dFvGet()$ntm)
```

Arguments

x	See reference
y	See reference
theta	See reference
wgt	See reference
cov	See reference
sigmai	See reference
tol	See reference
tau	See reference
itype	See reference
isigma	See reference
icnv	See reference
maxit	See reference
maxis	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.87

rysigm	<i>Iterative algorithm for the computation of an M-estimate of the scale parameter when the residuals are given</i>
--------	---

Description

See Marazzi A. (1993), p.94

Usage

```
rysigm(rs, wgt, exchi = chi, sigmai, np, tol = .dFvGet()$tlo,
       itype = .dFvGet()$ite, isigma = .dFvGet()$isg,
       maxis = .dFvGet()$mxt)
```

Arguments

rs	See reference
wgt	See reference
exchi	See reference
sigmai	See reference
np	See reference
tol	See reference
itype	See reference
isigma	See reference
maxis	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.94

 rywalg

W-algorithm for M-estimates

Description

See Marazzi A. (1993), p.87

Usage

```
rywalg(x, y, theta, wgt, cov, psp0 = psp(0), expsi = psi, exchi = chi,
       exrho = rho, sigmai, tol = .dFvGet()$tlo, gam = .dFvGet()$gma,
       tau = .dFvGet()$tua, itype = .dFvGet()$ite, isigma = .dFvGet()$isg,
       icnv = .dFvGet()$icn, maxit = .dFvGet()$mxt, maxis = .dFvGet()$mxs,
       nitmon = .dFvGet()$ntm)
```

Arguments

x	See reference
y	See reference
theta	See reference
wgt	See reference
cov	See reference
psp0	See reference
expsi	See reference
exchi	See reference
exrho	See reference
sigmai	See reference
tol	See reference
gam	See reference
tau	See reference
itype	See reference
isigma	See reference
icnv	See reference
maxit	See reference
maxis	See reference
nitmon	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.87

scal	<i>Scales a vector by a constant</i>
------	--------------------------------------

Description

See Marazzi A. (1993), p.349

Usage

```
scal(x, sa, n = nrow(x), incx = 1)
```

Arguments

x	See reference
sa	See reference
n	See reference
incx	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.349

scald	<i>Scales a double precision vector by a constant</i>
-------	---

Description

See Marazzi A. (1993), p.349

Usage

```
scald(x, sa, n = nrow(x), incx = 1)
```

Arguments

x	See reference
sa	See reference
n	See reference
incx	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.349

See Also

[scal](#)

srt1 *Sorts the components of a vector in ascending order*

Description

See Marazzi A. (1993), p.361

Usage

srt1(a, k1 = 1, k2 = n)

Arguments

a	See reference
k1	See reference
k2	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.361

See Also

[srt2](#)

srt2 *Sorts the components of a vector in ascending order and permutes the components of another vector accordingly*

Description

See Marazzi A. (1993), p.362

Usage

srt2(a, b, k1 = 1, k2 = n)

Arguments

a	See reference
b	See reference
k1	See reference
k2	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.362

See Also

[srt1](#)

swap	<i>Interchanges two vectors</i>
------	---------------------------------

Description

See Marazzi A. (1993), p.363

Usage

```
swap(x, y, n = nrow(x), incx = 1, incy = 1)
```

Arguments

x	See reference
y	See reference
n	See reference
incx	See reference
incy	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.363

See Also[swapd](#)

`swapd`*Interchanges two vectors (double precision)*

Description

See Marazzi A. (1993), p.363

Usage

```
swapd(x, y, n = nrow(x), incx = 1, incy = 1)
```

Arguments

<code>x</code>	See reference
<code>y</code>	See reference
<code>n</code>	See reference
<code>incx</code>	See reference
<code>incy</code>	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.363

See Also[swap](#)

tauare	<i>Asymptotic relative efficiency of the tau-test</i>
--------	---

Description

See Marazzi A. (1993), p.190

Usage

```
tauare(itype = .dFvGet()$ite, mu, maxit = .dFvGet()$mxe, cpsi, bb,  
       sigmax = 1, upper = .dFvGet()$upr, til = .dFvGet()$tli,  
       tol = .dFvGet()$tlo)
```

Arguments

itype	See reference
mu	See reference
maxit	See reference
cpsi	See reference
bb	See reference
sigmax	See reference
upper	See reference
til	See reference
tol	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.190

tfrn2t	<i>Computes the Rn2-test statistic for a linear hypothesis in canonical form</i>
--------	--

Description

See Marazzi A. (1993), p.187

Usage

```
tfrn2t(cov, theta, n, nq, tau = .dFvGet())$tua)
```

Arguments

cov	See reference
theta	See reference
n	See reference
nq	See reference
tau	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.187

tftaut	<i>Computes the tau-test statistic for a linear hypothesis in canonical form</i>
--------	--

Description

See Marazzi A. (1993), p.182

Usage

```
tftaut(rs1, rs2, wgt, exrho = rho, np, nq, sigma, itype = .dFvGet())$ite)
```

Arguments

rs1	See reference
rs2	See reference
wgt	See reference
exrho	See reference
np	See reference
nq	See reference
sigma	See reference
itype	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.182

tisrtc	<i>Permutes the columns of the design matrix: Predictors in omega are placed in the first q positions</i>
--------	---

Description

See Marazzi A. (1993), p.188

Usage

```
tisrtc(x, iv, n = nrow(x))
```

Arguments

x	See reference
iv	See reference
n	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.188

to.character	<i>Convert local variable to Fortran character</i>
--------------	--

Description

Function only needed for the interface

Usage

```
to.character(x)
```

Arguments

x	An R object
---	-------------

Value

x converted to character

to.double	<i>Convert local variable to Fortran double precision</i>
-----------	---

Description

Function only needed for the interface

Usage

```
to.double(x)
```

Arguments

x	An R numeric object
---	---------------------

Value

x converted to double precision

to.integer	<i>Convert local variable to Fortran integer</i>
------------	--

Description

Function only needed for the interface

Usage

```
to.integer(x)
```

Arguments

x An R numeric object

Value

x converted to integer

to.single	<i>Convert local variable to Fortran single precision</i>
-----------	---

Description

Function only needed for the interface

Usage

```
to.single(x)
```

Arguments

x An R numeric object

Value

x converted to single precision

tquant	<i>Inverse of the cumulative t-distribution function</i>
--------	--

Description

See Marazzi A. (1993), p.378

Usage

```
tquant(p, ifn)
```

Arguments

p	See reference
ifn	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.378

ttaskt	<i>Computes the matrix Ktau</i>
--------	---------------------------------

Description

See Marazzi A. (1993), p.184

Usage

```
ttaskt(cov, ainv, np, nq, mdc = np - nq, fact = .dFvGet())$ffc)
```

Arguments

cov	See reference
ainv	See reference
np	See reference
nq	See reference
mdc	See reference
fact	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.184

tteign

Computes the eigenvalues of the matrix Ktau

Description

See Marazzi A. (1993), p.186

Usage

```
tteign(covtau, nq, mdc = np - nq)
```

Arguments

covtau	See reference
nq	See reference
mdc	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.186

ugl *ub weight function for M-estimates in GLM*

Description

See Marazzi A. (1993), p.331

Usage

ugl(upar, npar = 4, s)

Arguments

upar	See reference
npar	See reference
s	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.331

Upcv *u' weight function for covariances*

Description

See Marazzi A. (1993), p.325

Usage

Upcv(svals)

Arguments

svals	A vector of input values
-------	--------------------------

Value

The values of the u function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.139

userfs *Dummy u user function*

Description

See Marazzi A. (1993), p.139

Usage

userfs(s)

Arguments

s A scalar input value

Value

The single precision value of the u function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.139

vcv *v weight function for covariances*

Description

See Marazzi A. (1993) p.327

Usage

vcv(s)

Arguments

s A scalar input value

Value

The value of the v function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.327

vpcv	<i>v' weight function for covariances</i>
------	---

Description

See Marazzi A. (1993), p.327

Usage

vpcv(s)

Arguments

s A scalar input value

Value

The value of the v' function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.327

Wcv	<i>w weight function for covariances</i>
-----	--

Description

See Marazzi A. (1993), p.328

Usage

Wcv(svals)

Arguments

svals A vector of input values

Value

The values of the w function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.328

wcv	<i>v weight function for covariances</i>
-----	--

Description

See Marazzi A. (1993), p.328

Usage

wcv(s)

Arguments

s A scalar input value

Value

The value of the w function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.328

wfshat	<i>Schweppe original weight proposal</i>
--------	--

Description

See Marazzi A. (1993), p.137

Usage

wfshat(xt, n = nrow(xt), sh)

Arguments

xt See reference
n See reference
sh See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.137

wimedv	<i>Initial value of the matrix A</i>
--------	--------------------------------------

Description

See Marazzi A. (1993), p.119

Usage

```
wimedv(x, nobs = nrow(x), itypw = .dFvGet()$itw,  
       init = .dFvGet()$ini, nfirst = nobs)
```

Arguments

x	See reference
nobs	See reference
itypw	See reference
init	See reference
nfirst	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.119

wpcv	<i>w' weight function for covariances</i>
------	---

Description

See Marazzi A. (1993), p.329

Usage

wpcv(svals)

Arguments

svals A vector of input values

Value

The values of the w' function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.329

wpcv	<i>w' weight function for covariances</i>
------	---

Description

See Marazzi A. (1993), p.329

Usage

wpcv(s)

Arguments

s A scalar input value

Value

The value of the w' function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.329

Www *w weight function for covariances*

Description

See Marazzi A. (1993), p.330

Usage

Www(svals)

Arguments

svals A vector of input values

Value

The values of the w weight function for each element of svals

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.330

www *w weight function*

Description

See Marazzi A. (1993), p.330

Usage

www(s)

Arguments

s A scalar input value

Value

The value of the w weight function for s

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.330

 wyfalg

Fixed-point algorithm for the computation of the matrix A

Description

See Marazzi A. (1993), p.121

Usage

```
wyfalg(x, a, gwt, exu = ucv, nobs = nrow(x), nvarq = 0,
      tau = .dFvGet()$tua, maxit = .dFvGet()$mxf, nitmon = .dFvGet()$ntm,
      icnv = .dFvGet()$icv, itypw = .dFvGet()$itw, igwt = 0,
      tol = .dFvGet()$tlo)
```

Arguments

x	See reference
a	See reference
gwt	See reference
exu	See reference
nobs	See reference
nvarq	See reference
tau	See reference
maxit	See reference
nitmon	See reference
icnv	See reference
itypw	See reference
igwt	See reference
tol	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.121

wyfcol	<i>Modified fixed-point algorithm for collinear data in the standardized case</i>
--------	---

Description

See Marazzi A. (1993), p.87

Usage

```
wyfcol(x, exu = ucV, nobs = nrow(x), iwgt = .dFvGet()$iwg,  
      apar = .dFvGet()$apr, tau = .dFvGet()$tua,  
      tol = .dFvGet()$tlo, maxit = .dFvGet()$mxf,  
      nitmon = .dFvGet()$ntm, icnv = .dFvGet()$icv)
```

Arguments

x	See reference
exu	See reference
nobs	See reference
iwgt	See reference
apar	See reference
tau	See reference
tol	See reference
maxit	See reference
nitmon	See reference
icnv	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.8

wygalg

Conjugate gradient algorithm for the computation of the lower-triangular matrix A in the standardized case

Description

See Marazzi A. (1993), p.127

Usage

```
wygalg(x, a, exu = ucV, exup = upcV, nobs = nrow(x),  
       maxit = .dFvGet()$mxg, nitmon = .dFvGet()$ntm,  
       icnv = .dFvGet()$icv, tol = .dFvGet()$tlo,  
       xfud = .dFvGet()$xfd)
```

Arguments

x	See reference
a	See reference
exu	See reference
exup	See reference
nobs	See reference
maxit	See reference
nitmon	See reference
icnv	See reference
tol	See reference
xfud	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.127

wynalg	<i>Newton-Huber algorithm for the computation of the lower-triangular matrix A in the standardized case</i>
--------	---

Description

See Marazzi A. (1993), p.87

Usage

```
wynalg(x, a, exu = ucV, exup = upcV, nobs = nrow(x),  
       maxit = .dFvGet()$mxn, nitmon = .dFvGet()$ntm,  
       icnv = .dFvGet()$icv, tol = .dFvGet()$tlo,  
       xfud = .dFvGet()$xfd)
```

Arguments

x	See reference
a	See reference
exu	See reference
exup	See reference
nobs	See reference
maxit	See reference
nitmon	See reference
icnv	See reference
tol	See reference
xfud	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.87

xerf

Gaussian density function

Description

See Marazzi A. (1993), p.369

Usage

xerf(kode = 2, x)

Arguments

kode	See reference
x	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.369

xerp

Density of the norm of a standard Gaussian vector with p components

Description

See Marazzi A. (1993), p.370

Usage

xerp(ip, xlcnst = -1, s)

Arguments

ip	See reference
xlcnst	See reference
s	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.370

xsy *Evaluates a quadratic form*

Description

See Marazzi A. (1993), p.352

Usage

xsy(x, y, s)

Arguments

x	See reference
y	See reference
s	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.352

xsyd *Evaluates a quadratic form (double precision)*

Description

See Marazzi A. (1993) p.352

Usage

xsyd(x, y, s)

Arguments

x	See reference
y	See reference
s	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.352

See Also

[xsy](#)

zemll

Zeng method for censored data

Description

See Reference

Usage

zemll(b, x, yo, do)

Arguments

b	See reference
x	See reference
yo	See reference
do	See reference

Value

See reference

References

Marazzi A. (1993) *Algorithm, Routines, and S functions for Robust Statistics*. Wadsworth & Brooks/cole, Pacific Grove, California. p.216

Index

- * **Stats**
 - zemll, 154
- * **algebra**
 - mchl, 83
 - mchld, 84
 - mhat, 88
 - permc, 104
 - permv, 104
 - rimtrd, 118
 - rimtrf, 119
 - rmvc, 120
 - scal, 127
 - scald, 128
 - tteign, 138
- * **distribution**
 - binprd, 31
 - cerf, 31
 - cerfd, 32
 - chisq, 34
 - cquant, 39
 - fcum, 49
 - gauss, 51
 - gaussd, 51
 - ingama, 64
 - liinds, 75
 - poissn, 105
 - probst, 107
 - quant, 111
 - Random, 112
 - ruben, 120
 - tquant, 137
- * **interface**
 - messagena, 84
 - to.character, 135
 - to.double, 135
 - to.integer, 136
 - to.single, 136
- * **misc**
 - precd, 106
 - precs, 106
- * **package**
 - robeth-package, 5
- * **robust**
 - airef0, 29
 - airefq, 30
 - Chi, 33
 - chi, 33
 - cia2b2, 35
 - cibeat, 35
 - cicloc, 36
 - cifact, 37
 - cimedv, 37
 - cirock, 38
 - cyfalg, 40
 - cygalg, 41
 - cynalg, 42
 - Dbinom, 43
 - dpoiss, 47
 - Fn.Exp.f, 50
 - gintac, 53
 - gyastp, 55
 - gycstp, 56
 - gymain, 56
 - gytstp, 58
 - hylmse, 60
 - hylvse, 61
 - hysest, 62
 - hysestw, 63
 - kfascv, 64
 - kfedcb, 65
 - kfedcc, 66
 - kffacv, 66
 - kiascv, 67
 - kiedch, 68
 - kiedcu, 68
 - ktaskv, 69
 - ktaskw, 70
 - libet0, 71

libeth, 72
 libetu, 72
 liepsh, 73
 liepsu, 74
 lrfctd, 78
 lyhalg, 79
 lytau2, 81
 lywalg, 82
 mfragr, 86
 mirtsr, 90
 myhbhe, 101
 mymvlm, 101
 Psi, 107
 psi, 108
 Psp, 108
 psp, 109
 QD2coef.f, 109
 QD2funC.f, 110
 Qn.Exp.f, 111
 Regtau.f, 113
 RegtauW.f, 113
 Rho, 114
 rho, 115
 ribet0, 115
 ribeth, 116
 ribetu, 116
 riclls, 117
 rilars, 118
 robeth-package, 5
 rybifr, 121
 ryhalg, 122
 rynalg, 123
 rysalg, 124
 rysigm, 125
 rywalg, 126
 tauare, 132
 tfrn2t, 133
 tftaut, 133
 tisrtc, 134
 ttaskt, 137
 Ucv, 139
 ucv, 139
 ugl, 140
 Upcv, 140
 upcv, 141
 userfd, 141
 userfs, 142
 vcv, 142
 vpcv, 143
 Wcv, 143
 wcv, 144
 wfshat, 144
 wimedv, 145
 Wpcv, 146
 wpcv, 146
 Www, 147
 www, 147
 wyfalg, 148
 wyfcol, 149
 wygalg, 150
 wynalg, 151
 * **stats**
 glmdev, 54
 * **univar**
 lywalg, 82
 * **utilities**
 addc, 28
 cfrcov, 32
 comval, 39
 dfcomn, 43
 dfrpar, 45
 dfvals, 45
 dotp, 46
 dotpd, 46
 exch, 48
 exchd, 48
 fstord, 50
 gfedca, 52
 h12, 59
 h12d, 59
 lgama, 71
 liclls, 73
 liindh, 74
 liindw, 76
 lilars, 76
 littst, 77
 lmdd, 77
 lyhdle, 80
 lymnwt, 80
 mff, 85
 mffd, 85
 mfy, 87
 mfyd, 87
 minv, 89
 minvd, 89
 mly, 91

- mlyd, 91
 - msf, 92
 - msf1, 93
 - msf1d, 93
 - msfd, 94
 - mss, 94
 - mssd, 95
 - mtt1, 96
 - mtt1d, 96
 - mtt2, 97
 - mtt2d, 97
 - mtt3, 98
 - mtt3d, 99
 - mty, 99
 - mtyd, 100
 - nlgm, 102
 - nrm2, 103
 - nrm2d, 103
 - srt1, 129
 - srt2, 129
 - swap, 130
 - swapd, 131
 - xerf, 152
 - xerp, 152
 - xsy, 153
 - xsyd, 153
- addc, 28
- airef0, 29
- airefq, 30
- binprd, 31
- cerf, 31
- cerfd, 32
- cfrcov, 32
- Chi, 33
- chi, 33
- chisq, 34
- cia2b2, 35
- cibeat, 35
- cicloc, 36
- cifact, 37
- cimedv, 37
- cirock, 38
- comval, 39
- cquant, 39
- cyfalg, 40
- cygalg, 41
- cynalg, 42
- Dbinom, 43
- dfcomn, 43
- dfrpar, 45
- dfvals, 45
- dotp, 46
- dotpd, 46
- dpoiss, 47
- exch, 48
- exchd, 48
- fcum, 49
- Fn.Exp.f, 50
- fstord, 50
- gauss, 51
- gaussd, 51
- gfedca, 52
- gintac, 53
- glmdev, 54
- gyastp, 55
- gycstp, 56
- gymain, 56
- gytstp, 58
- h12, 59
- h12d, 59
- hylmse, 60
- hyltse, 61
- hysest, 62
- hysestw, 63
- ingama, 64
- kfascv, 64
- kfedcb, 65
- kfedcc, 66
- kffacv, 66
- kiascv, 67
- kiedch, 68
- kiedcu, 68
- ktaskv, 69
- ktaskw, 70
- lgama, 71
- libet0, 71
- libeth, 72
- libetu, 72

- liclls, 73
- liepsh, 73
- liepsu, 74
- liindh, 74
- liinds, 75
- liindw, 76
- lilars, 76
- littst, 77
- lmdd, 77
- lrfctd, 78
- lyhalg, 79
- lyhdle, 80
- lymwt, 80
- lytau2, 81
- lywalg, 82

- mchl, 83
- mchld, 84
- messagena, 84
- mff, 85
- mffd, 85
- mfragr, 86
- mfy, 87
- mfyd, 87
- mhat, 88
- minv, 89
- minvd, 89
- mirtsr, 90
- mly, 91
- mlyd, 91
- msf, 92
- msf1, 93
- msf1d, 93
- msfd, 94
- mss, 94
- mssd, 95
- mtt1, 96
- mtt1d, 96
- mtt2, 97
- mtt2d, 97
- mtt3, 98
- mtt3d, 99
- mty, 99
- mtyd, 100
- myhbhe, 101
- mymv1m, 101

- nlgm, 102
- nrm2, 103

- nrm2d, 103

- permc, 104
- permv, 104
- poissn, 105
- precd, 106
- precs, 106
- probst, 107
- Psi, 107
- psi, 108
- Psp, 108
- psp, 109

- QD2coef.f, 109
- QD2funC.f, 110
- Qn.Exp.f, 111
- quant, 111

- Random, 112
- Regtau.f, 113
- RegtauW.f, 113
- Rho, 114
- rho, 115
- ribet0, 115
- ribeth, 116
- ribetu, 116
- riclls, 117
- rilars, 118
- rimtrd, 118
- rimtrf, 119, 119
- rmvc, 120
- robeth (robeth-package), 5
- robeth-package, 5
- ruben, 120
- rybifr, 121
- ryhalg, 122
- rynalg, 123
- rysalg, 124
- rysigm, 125
- rywalg, 126

- scal, 127, 128
- scald, 128
- srt1, 129, 130
- srt2, 129, 129
- swap, 130, 131
- swapd, 131, 131

- tauare, 132

tfrn2t, 133
tftaut, 133
tisrtc, 134
to.character, 135
to.double, 135
to.integer, 136
to.single, 136
tquant, 137
ttaskt, 137
tteign, 138

Ucv, 139
ucv, 139
ugl, 140
Upcv, 140
upcv, 141
userfd, 141
userfs, 142

vcv, 142
vpcv, 143

Wcv, 143
wcv, 144
wfshat, 144
wimedv, 145
Wpcv, 146
wpcv, 146
Www, 147
www, 147
wyfalg, 148
wyfcol, 149
wygalg, 150
wynalg, 151

xerf, 152
xerp, 152
xsy, 153, 154
xsyd, 153

zemll, 154