Package ‘robustX’

June 16, 2023

Type Package
Title 'eXtra' / 'eXperimental' Functionality for Robust Statistics
Version 1.2-7
Date 2023-06-14
Maintainer Martin Maechler <maechler@stat.math.ethz.ch>

Description Robustness -- 'eXperimental', 'eXtraneous', or 'eXtraordinary'
Functionality for Robust Statistics. Hence methods which are not well established,
often related to methods in package 'robustbase'. Amazingly, 'BACON()', originally by
has become established in places. The "barrow wheel" `rbwheel()` is from

Imports grDevices, graphics, stats, utils, robustbase (>= 0.92-3)
Suggests MASS, lattice, pcaPP
Enhances ICS
License GPL (>= 2)
Encoding UTF-8

Author Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>),
Werner A. Stahel [aut],
Rolf Turner [ctb] (reclas()),
Ueli Oetliker [ctb] (original version of BACON() and mvBACON for S+),
Tobias Schoch [ctb] (init.sel="V2" for BACON; fix alpha)

Repository CRAN
Date/Publication 2023-06-16 07:30:02 UTC

R topics documented:

robustX-package .......................................................... 2
BACON ................................................................. 3
covNNC ............................................................... 6
L1median ................................................................. 7
Description

The package robustX aims to be a collection of \textit{R} functionality for robust statistics of methods and ideas that are considered as proposals, experimental, for experiences or just too much specialized to be part of the “Robust Basics” package robustbase.

Details

Package: robustX
Type: Package
Title: 'eXtra' / 'eXperimental' Functionality for Robust Statistics
Version: 1.2-7
Date: 2023-06-14
Authors@R: c(person("Martin","Maechler", role="aut","cre"), email="maechler@stat.math.ethz.ch", comment = c(ORCID = "0000-0002-8685-9910")), person("Tobias","Schoch", role="ctb", comment = "init.sel="V2" for BACON; fix alpha")
Maintainer: Martin Maechler <maechler@stat.math.ethz.ch>
Imports: grDevices, graphics, stats, utils, robustbase (>= 0.92-3)
Suggests: MASS, lattice, pcaPP
Enhances: ICS
License: GPL (>= 2)
Encoding: UTF-8
Author: Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>), Werner A. Stuhel [aut], Rolf Turner [ctb]

Index of help topics:

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACON</td>
<td>BACON for Regression or Multivariate Covariance Estimation</td>
</tr>
<tr>
<td>L1median</td>
<td>Compute the Multivariate L1-Median aka 'Spatial Median'</td>
</tr>
<tr>
<td>Qrot</td>
<td>Rotation Matrix to Specific Direction</td>
</tr>
<tr>
<td>covNNC</td>
<td>Robust Covariance Estimation via Nearest Neighbor Cleaning</td>
</tr>
<tr>
<td>mvBACON</td>
<td>BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators</td>
</tr>
<tr>
<td>rbwheel</td>
<td>Multivariate Barrow Wheel Distribution Random Vectors</td>
</tr>
</tbody>
</table>
### Description

BACON, short for ‘Blocked Adaptive Computationally-Efficient Outlier Nominators’, is a somewhat robust algorithm (set), with an implementation for regression or multivariate covariance estimation.

BACON() applies the multivariate (covariance estimation) algorithm, using mvBACON(x) in any case, and when y is not NULL adds a regression iteration phase, using the auxiliary .lmBACON() function.

### Usage

```r
BACON(x, y = NULL, intercept = TRUE,
       m = min(collect * p, n * 0.5),
       init.sel = c("Mahalanobis", "dUniMedian", "random", "manual", "V2"),
       man.sel, init.fraction = 0, collect = 4,
       alpha = 0.05, alphaLM = alpha, maxsteps = 100, verbose = TRUE)

# *Auxiliary* function:
.lmBACON(x, y, intercept = TRUE,
       init.dis, init.fraction = 0, collect = 4,
       alpha = 0.05, maxsteps = 100, verbose = TRUE)
```

### Examples

```r
pairs( rbwheel(100, 4) )
```
Arguments

\texttt{x} \hspace{1cm} \text{a multivariate matrix of dimension \([n \times p]\) considered as containing no missing values.}

\texttt{y} \hspace{1cm} \text{the response (n vector) in the case of regression, or NULL for the multivariate case, where just \texttt{mvBACON()} is returned.}

\texttt{intercept} \hspace{1cm} \text{logical indicating if an intercept has to be used for the regression.}

\texttt{m} \hspace{1cm} \text{integer in 1:n specifying the size of the initial basic subset; used only when init.sel is not "manual"; see \texttt{mvBACON}.}

\texttt{init.sel} \hspace{1cm} \text{character string, specifying the initial selection mode; see \texttt{mvBACON}.}

\texttt{man.sel} \hspace{1cm} \text{only when init.sel == "manual", the indices of observations determining the initial basic subset (and m <- length(man.sel)).}

\texttt{init.dis} \hspace{1cm} \text{the distances of the x matrix used for the initial subset determined by \texttt{mvBACON}.}

\texttt{init.fraction} \hspace{1cm} \text{if this parameter is > 0 then the tedious steps of selecting the initial subset are skipped and an initial subset of size n * init.fraction is chosen (with smallest dis).}

\texttt{collect} \hspace{1cm} \text{numeric factor chosen by the user to define the size of the initial subset (p * collect).}

\texttt{alpha} \hspace{1cm} \text{number in (0, 1) determining the cutoff value for the Mahalanobis distances (multivariate outlier nomination in \texttt{mvBACON()}), or the discrepancies for regression, see \texttt{alphaLM}.}

\texttt{alphaLM} \hspace{1cm} \text{number in (0, 1) where a 1-alphaM t-quantile is the cutoff for the discrepancies (for regression, \texttt{.lmBACON()}); see details.}

\texttt{maxsteps} \hspace{1cm} \text{the maximal number of iteration steps (to prevent infinite loops).}

\texttt{verbose} \hspace{1cm} \text{logical indicating if messages are printed which trace progress of the algorithm.}

Details

Notably about the initial selection mode, \texttt{init.sel}, see its description in the \texttt{mvBACON} arguments list.

The choice of \texttt{alpha} and \texttt{alphaLM}:

- Multivariate outlier nomination: see the Details section of \texttt{mvBACON}.

- Regression: Let \(t_r(\alpha)\) denote the \(1 - \alpha\) quantile of the Student \(t\)-distribution with \(r\) degrees of freedom, where \(r\) is the number of elements in the current subset; e.g., \(t_r(0.05)\) is the 0.95 quantile. Following Billor et al. (2000), the cutoff value for the discrepancies is defined as \(t_r(\alpha/(2r + 2))\), and they use \(\alpha = 0.05\). Note that this is argument \texttt{alphaLM} (defaulting to \texttt{alpha}) for \texttt{BACON}.

Value

\texttt{BACON(x,y,\ldots)} for regression returns a \texttt{list} with components

\texttt{subset} \hspace{1cm} \text{the observation indices (in 1:n) denoting a subset of “good” supposedly outlier-free observations.
tis the \( t_i(y_m, X_m) \) of eq (6) in the reference; the clean “basic subset” in the algorithm is defined the observations \( i \) with the smallest \( |t_i| \), and the \( t_i \) can be regarded as scaled predicted errors.

mv.dis the (final) discrepancies or distances of mvBACON()

mv.subset the “good” subset from mvBACON(), used to start the regression iterations.

Note

“BACON” was also chosen in honor of Francis Bacon:

Whoever knows the ways of Nature will more easily notice her deviations; and, on the other hand, whoever knows her deviations will more accurately describe her ways.

Francis Bacon (1620), Novum Organum II 29.

Author(s)

Ueli Oetliker, Swiss Federal Statistical Office, for S-plus 5.1; 25.05.2001; modified six times till 17.6.2001.

Port to \( \mathbb{R} \), testing etc, by Martin Maechler. Daniel Weeks (at pitt.edu) proposed a fix to a long standing buglet in GiveTis() computing the \( t_i \), which was further improved Maechler, for robustX version 1.2-3 (Feb. 2019).

Correction of alpha default, from 0.95 to 0.05, by Tobias Schoch, see mvBACON.

References


See Also

mvBACON, the multivariate version of the BACON algorithm.

Examples

data(starsCYG, package = "robustbase")
## Plot simple data and fitted lines
plot(starsCYG)
lmST <- lm(log.light ~ log.Te, data = starsCYG)
abline(lmST, col = "gray") # least squares line
str(B.ST <- with(starsCYG, BACON(x = log.Te, y = log.light)))
## 'subset': A good set of of points (to determine regression):
colB <- adjustcolor(2, 1/2)
points(log.light ~ log.Te, data = starsCYG, subset = B.ST$subset,
       pch = 19, cex = 1.5, col = colB)
## A BACON-derived line:
lmB <- lm(log.light ~ log.Te, data = starsCYG, subset = B.ST$subset)
abline(lmB, col = colB, lwd = 2)
require(robustbase)
covNNC <- lmrob(log.light ~ log.Te, data = starsCYG)
abline(RlmST, col = "blue")

---

### covNNC

**Robust Covariance Estimation via Nearest Neighbor Cleaning**

**Description**

covNNC() estimates robust covariance/dispersion matrices by the nearest neighbor variance estimation (NNVE) or (rather) “Nearest Neighbor Cleaning” (NNC) method of Wang and Raftery (2002, *JASA*).

**Usage**

covNNC(X, k = min(12, n - 1), pnoise = 0.05, emconv = 0.001,
bound = 1.5, extension = TRUE, devsm = 0.01)

**Arguments**

- **X**: matrix in which each row represents an observation or point and each column represents a variable.
- **k**: desired number of nearest neighbors (default is 12)
- **pnoise**: percent of added noise
- **emconv**: convergence tolerance for EM
- **bound**: value used to identify surges in variance caused by outliers wrongly included as signal points (bound = 1.5 means a 50 percent increase)
- **extension**: whether or not to continue after reaching the last chi-square distance. The default is to continue, which is indicated by setting extension = TRUE.
- **devsm**: when extension = TRUE, the algorithm stops if the relative difference in variance is less than devsm. (default is 0.01)

**Value**

A list with components

- **cov**: covariance matrix
- **mu**: mean vector
- **postprob**: posterior probability
- **classification**: classification (0=noise otherwise 1) obtained by rounding postprob
- **innc**: list of initial nearest neighbor cleaning results (components are the covariance, mean, posterior probability and classification)
Note

Terms of use: GPL version 2 or newer.

MM: Even though covNNC() is backed by a serious scientific publication, I cannot recommend its use at all.

Author(s)

Naisyin Wang <nwang@stat.tamu.edu> and Adrian Raftery <raftery@stat.washington.edu>
with contributions from Chris Fraley <fraley@stat.washington.edu>.

covNNC(), then named cov.nnve(), used to be (the only function) in CRAN package covRobust (2003), which was archived in 2012.

Martin Maechler allowed ncol(X) == 1, sped up the original code, by reducing the amount of scaling; further, the accuracy was increased (using internal q.dDk()). The original version is available, unexported as robustX:::covNNC1.

References


See also University of Washington Statistics Technical Report 368 (2000); see at https://stat.uw.edu/research/tech-reports/

See Also

cov.mcd from package MASS; covMcd, and covOGK from package robustbase.

The whole package rrcov.

Examples

```r
data(iris)
covNNC(iris[-5])

data(hbk, package="robustbase")
hbk.x <- data.matrix(hbk[, 1:3])
covNNC(hbk.x)
```

---

L1median

Compute the Multivariate L1-Median aka 'Spatial Median'

---
Description

Compute the multivariate $L_1$-median $m$, also called “Spatial Median”, i.e., the minimizer of

$$\sum_{i=1}^n \|x_i - m\|,$$

where $\|u\| = \sqrt{\sum_{j=1}^p u_j^2}.$

As a convex problem, there’s always a global minimizer, computable not by a closed formula but rather an iterative search. As the (partial) first derivatives of the objective function is undefined at the data points, the minimization is not entirely trivial.

Usage

```r
L1median(X, m.init = colMeans(X), weights = NULL,
method = c("nlm", "HoCrJo", "VardiZhang", optimMethods, nlminbMethods),
pscale = apply(abs(centr(X, m.init)), 2, mean, trim = 0.40),
tol = 1e-08, maxit = 200, trace = FALSE,
zero.tol = 1e-15, ...)
```

Arguments

- **X**: numeric matrix of dimension $n \times p$, say.
- **m.init**: starting value for $m$; typically and by default the coordinatewise median.
- **weights**: optional numeric vector of non-negative weights; currently only implemented for method "VardiZhang".
- **method**: character string specifying the computational method, i.e., the algorithm to be used (can be abbreviated).
- **pscale**: numeric p-vector of positive numbers, the coordinate-wise scale (typical size of $\delta m_j$), where $m$ is the problem’s solution.
- **tol**: positive number specifying the (relative) convergence tolerance.
- **maxit**: positive integer specifying the maximal number of iterations (before the iterations are stopped prematurely if necessary).
- **trace**: an integer specifying the tracing level of the iterations; 0 does no tracing
- **zero.tol**: for method "VardiZhang", a small positive number specifying the tolerance for determining that the iteration is ‘exactly’ at a data point (which is a singularity).
- **...**: optional arguments to `nlm()` or the control (list) arguments of `optim()`, or `nlminb()`, respectively.

Details

Currently, we have to refer to the “References” below.

Value

 currently the result depends strongly on the method used.

FIXME. This will change considerably.
Author(s)

Martin Maechler. Method "HoCrJo" is mostly based on Kristel Joossens’ R function, implementing Hossjer and Croux (1995).

References


See Also

median, covMcd

CRAN package pcaPP added more L1 median methods, re-implementing our R versions in C++, see Fritz et al.(2012) and e.g., l1median_NLM().

Examples

data(stackloss)
L1median(stackloss)
L1median(stackloss, method = "HoCrJo")

## Explore all methods:
m <- eval(formals(L1median)$method); allMeths <- m[m != "Brent"]
L1m <- sapply(allMeths, function(meth) L1median(stackloss, method = meth))
## --> with a warning for L-BFGS-B
str(L1m)
pm <- sapply(L1m, function(.) if(is.numeric(.)) . else .$par)
t(pm) # SANN differs a bit; same objective ?

mvBACON

BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators

Description

This function performs an outlier identification algorithm to the data in the x array [n x p] and y vector [n] following the lines described by Hadi et al. for their BACON outlier procedure.
Usage

```r
mvBACON(x, collect = 4, m = min(collect * p, n * 0.5), alpha = 0.05,
        init.sel = c("Mahalanobis", "dUniMedian", "random", "manual", "V2"),
        man.sel, maxsteps = 100, allowSingular = FALSE, verbose = TRUE)
```

Arguments

- **x**: numeric matrix (of dimension \([n \times p]\)), not supposed to contain missing values.
- **collect**: a multiplication factor \(c\), when \(\text{init.sel}\) is not "manual", to define \(m\), the size of the initial basic subset, as \(m := c \cdot p\), in practice, \(m \leftarrow \min(p \cdot \text{collect}, n/2)\).
- **m**: integer in 1:n specifying the size of the initial basic subset; used only when \(\text{init.sel}\) is not "manual".
- **alpha**: determines the cutoff value for the Mahalanobis distances (see details).
- **init.sel**: character string, specifying the initial selection mode; implemented modes are:
  - "Mahalanobis" based on Mahalanobis distances (default); the version \(V_1\) of the reference; affine invariant but not robust.
  - "dUniMedian" based on the distances from the univariate medians; similar to the version \(V_2\) of the reference; robust but not affine invariant.
  - "random" based on a random selection, i.e., reproducible only via \text{set.seed()}.
  - "manual" based on manual selection; in this case, a vector \text{man.sel} containing the indices of the selected observations must be specified.
  - "V2" based on the Euclidean norm from the univariate medians; this is the version \(V_2\) of the reference; robust but not affine invariant.
  "Mahalanobis" and "V2" where proposed by Hadi and the other authors in the reference as versions ‘V_1’ and ‘V_2’, as well as "manual", while "random" is provided in order to study the behaviour of BACON. Option "dUniMedian" is similar to "V2" and is due to U. Oetliker.
- **man.sel**: only when \(\text{init.sel} = \text{"manual"}\), the indices of observations determining the initial basic subset (and \(m \leftarrow \text{length(man.sel)}\)).
- **maxsteps**: maximal number of iteration steps.
- **allowSingular**: logical indicating a solution should be sought also when no matrix of rank \(p\) is found.
- **verbose**: logical indicating if messages are printed which trace progress of the algorithm.

Details

Remarks on the tuning parameter \(\alpha\): Let \(\chi_p^2\) be a chi-square distributed random variable with \(p\) degrees of freedom (\(p\) is the number of variables; \(n\) is the number of observations). Denote the \((1 - \alpha)\) quantile by \(\chi_p^2(\alpha)\), e.g., \(\chi_p^2(0.05)\) is the 0.95 quantile. Following Billor et al. (2000), the cutoff value for the Mahalanobis distances is defined as \(\chi_p^2(\alpha/n)\) (the square root of \(\chi_p^2\)) times a correction factor \(c(n, p)\), \(n\) and \(p\), and they use \(\alpha = 0.05\).
Value

- **subset**: logical vector of length \( n \) where the \( i \)-th entry is true iff the \( i \)-th observation is part of the final selection.
- **dis**: numeric vector of length \( n \) with the (Mahalanobis) distances.
- **cov**: \( p \times p \) matrix, the corresponding robust estimate of covariance.

Author(s)

Ueli Oetliker, Swiss Federal Statistical Office, for S-plus 5.1. Port to \( \mathbb{R} \), testing etc, by Martin Maechler; Init selection "V2" and correction of default alpha from 0.95 to 0.05, by Tobias Schoch, FHNW Olten, Switzerland.

References


See Also

covMcd for a high-breakdown (but more computer intensive) method; BACON for a "generalization", notably to regression.

Examples

```r
require(robustbase) # for example data and covMcd():
## simple 2D example:
plot(starsCYG, main = "starsCYG data (n=47)"
B.st <- mvBACON(starsCYG)
points(starsCYG[!B.st$subset,], pch = 4, col = 2, cex = 1.5)
stopifnot(identical(which(!B.st$subset), c(7L, 11L, 20L, 30L, 34L))
## finds the 4 clear outliers (and 1 "borderline");
## it does not find obs. 14 which is an outlier according to covMcd(.)
iniS <- setNames(, eval(formals(mvBACON)$init.sel)) # all initialization methods, incl "random" set.seed(123)
Bs.st <- lapply(iniS[iniS != "manual"], function(s)
  mvBACON(as.matrix(starsCYG), init.sel = s, verbose=FALSE))
ii <- -match("steps", names(Bs.st[[1]]))
Bs.si <- lapply(Bs.st, `[`, ii)
stopifnot(exprs = {
  length(Bs.si) >= 4
  length(unique(Bs.si)) == 1 # all 4 methods give the same
})
## Example where "dUniMedian" and "V2" differ:
data(pulpfiber, package="robustbase")
dU.plp <- mvBACON(as.matrix(pulpfiber), init.sel = "dUniMedian")
V2.plp <- mvBACON(as.matrix(pulpfiber), init.sel = "V2")
```
```r
(oU <- which(! dU.plp$subset))
(o2 <- which(! V2.plp$subset))
stopifnot(setdiff(o2, oU) %in% c(57L,58L,59L,62L))
## and 57, 58, 59, and 62 *are* outliers according to covMcd(.)
##
## 'coleman' from pkg 'robustbase'
coleman.x <- data.matrix(coleman[, 1:6])
Cc <- covMcd (coleman.x) # truly robust
summary(Cc) # -> 6 outliers (1,3,10,12,17,18)
Cb1 <- mvBACON(coleman.x) ##-> subset is all TRUE hmm??
Cb2 <- mvBACON(coleman.x, init.sel = "dUniMedian")
stopifnot(all.equal(Cb1, Cb2))
## try 20 different random starts:
Cb.r <- lapply(1:20, function(i) { set.seed(i)
  mvBACON(coleman.x, init.sel="random", verbose=FALSE) })
nm <- names(Cb.r[[1]]); nm <- nm[nm != "steps"]
all(eqC <- sapply(Cb.r[-1], function(CC) all.equal(CC[nm], Cb.r[[1]][nm]))) # TRUE
## --> BACON always breaks down, i.e., does not see the outliers here
## breaks down even when manually starting with all the non-outliers:
Cb.man <- mvBACON(coleman.x, init.sel = "manual",
  man.sel = setdiff(1:20, c(1,3,10,12,17,18)))
which( ! Cb.man$subset) # the outliers according to mvBACON : _none_
```

---

**Qrot**

**Rotation Matrix to Specific Direction**

**Description**

Construct the $p \times p$ rotation matrix that rotates the unit vector $(1,0,...,0)$, i.e., the $x_1$-axis, onto $(1,1,1,...,1)/\sqrt{p}$, or more generally to $u/\|u\|$ ($u := \text{unit.image}$).

**Usage**

```r
Qrot(p, transpose = FALSE, unit.image = rep(1, p))
```

**Arguments**

- `p` integer; the dimension (of the vectors involved).
- `transpose` logical indicating if the transposed matrix is to returned.
- `unit.image` numeric vector of length $p$ onto which the unit vector should be rotated; defaults to “the diagonal” $\propto(1,1,1,...,1)$.

**Details**

The qr decomposition is used for a Gram-Schmitt basis orthogonalization.

**Value**

$p \times p$ orthogonal matrix which rotates $(1,0,...,0)$ onto a vector proportional to `unit.image`. 
Author(s)

Martin Maechler

See Also

qr, matrix (and vector) multiplication, %*%.

Examples

Q <- Qrot(6)
zapsmall(crossprod(Q)) # 6 x 6 unity <==> Q'Q = I <==> Q orthogonal

if(require("MASS")) {
  Qt <- Qrot(6, transpose = TRUE)
  stopifnot(all.equal(Qt, t(Q)))
  fractions(Qt ^2) # --> 1/6 1/30 etc, in an almost lower-triangular matrix
}

rbwheel Multivariate Barrow Wheel Distribution Random Vectors

Description

Generate $p$-dimensional random vectors according to Stahel’s Barrow Wheel Distribution.

Usage

rbwheel(n, p, frac = 1/p, sig1 = 0.05, sig2 = 1/10,
    rGood = rnorm,
    rOut = function(n) sqrt(rchisq(n, p - 1)) * sign(runif(n, -1, 1)),
    U1 = rep(1, p),
    scaleAfter = TRUE, scaleBefore = FALSE, spherize = FALSE,
    fullResult = FALSE)

Arguments

n integer, specifying the sample size.

p integer, specifying the dimension (aka number of variables).

frac numeric, the proportion of outliers. The default, $1/p$, corresponds to the (asymptotic) breakdown point of M-estimators.

sig1 thickness of the “wheel”, ($= \sigma \text{(good[,1])}$), a non-negative numeric.

sig2 thickness of the “axis” (compared to 1).

rGood function; the generator for “good” observations.

rOut function, generating the outlier observations.

U1 p-vector to which $(1, 0, \ldots, 0)$ is rotated.
scaleAfter logical indicating if the matrix is re-scaled after rotation (via `scale()`). Default TRUE; note that this used to be false by default in the first public version.

scaleBefore logical indicating if the matrix is re-scaled before rotation (via `scale()`).

spherize logical indicating if the matrix is to be “spherized”, i.e., rotated and scaled to have empirical covariance $I_p$. This means that the principal components are used (before rotation).

fullResult logical indicating if in addition to the $n \times p$ matrix, some intermediate quantities are returned as well.

Details

....

Value

By default (when `fullResult` is FALSE), an $n \times p$ matrix of $n$ sample vectors of the $p$ dimensional barrow wheel distribution, with an attribute, `n1` specifying the exact number of “good” observations, $n_1 \approx (1 - f) \cdot n, f = \frac{\text{I}}{\text{II}}$.

If `fullResult` is TRUE, a list with components

X the $n \times p$ matrix of above, $X = X_0 \%*\% A$, where $A \leftarrow Qrot(p, u = U_1)$, and $X_0$ is the corresponding matrix before rotation, see below.

X0 ........

A the $p \times p$ rotation matrix, see above.

n1 the number of “good” observations, see above.

n2 the number of “outlying” observations, $n_2 = n - n_1$.

Author(s)

Werner Stahel and Martin Maechler

References

http://stat.ethz.ch/people/maechler/robustness


Examples

```
set.seed(17)
rX8 <- rbwheel(1000,8, fullResult = TRUE, scaleAfter=FALSE)
with(rX8, stopifnot(all.equal(X, X0 %*% A, tol = 1e-15),
    all.equal(X0, X %*% t(A), tol = 1e-15)))
##--> here, don't need to keep X0 (nor A, since that is Qrot(p))

## for n = 100, you don't see "it", but may guess .. :
n <- 100
pairs(r <- rbwheel(n,6))
```
n1 <- attr(r,"n1") ; pairs(r, col=1+((1:n) > n1))

## for n = 500, you *do* see it :
 n <- 500
 pairs(r <- rbwheel(n,6))
## show explicitly
 n1 <- attr(r,"n1") ; pairs(r, col=1+((1:n) > n1))

## but increasing sig2 does help:
pairs(r <- rbwheel(n,6, sig2 = .2))

## show explicitly
 n1 <- attr(r,"n1") ; pairs(r, col=1+((1:n) > n1))

set.seed(12)
pairs(X <- rbwheel(n, 7, spherize=TRUE))
colSums(X) # already centered

if(require("ICS") && require("robustbase")) {
  # ICS: Compare M-estimate [Max.Lik. of t_{df = 2}] with high-breakdown :
  stopifnot(require("MASS"))
  X.paM <- ics(X, S1 = cov, S2 = function(.) cov.trob(.), nu=2)$cov, stdKurt = FALSE)
  X.paM.<- ics(X, S1 = cov, S2 = function(.) tM(.), df=2)$V, stdKurt = FALSE)
  X.paR <- ics(X, S1 = cov, S2 = function(.) covMcd(.)$cov, stdKurt = FALSE)
  plot(X.paM) # not at all clear
  plot(X.paM.)# ditto
  plot(X.paR)# very clear
}

## Similar such experiments ---> demo(rbwheel_d) and demo(rbwheel_ics)

---

reclas

**Recursive Robust Median-like Location and Scale**

**Description**

Calculate an estimate of location, asymptotically equivalent to the median, and an estimate of scale equal to the MEAN absolute deviation. Both done recursively.

**Usage**

reclas(y, b = 0.2, mfn = function(n) 0.1 * n^(-0.25),
nstart = 30, m0 = median(y0),
scon=NULL, updateScale = is.null(scon))
Arguments

- **y**: numeric vector of i.i.d. data whose location and scale parameters are to be estimated.
- **b**: numeric tuning parameter (default value equal to that used by Holst, 1987).
- **mfn**: a function of the index of the data which must be positive and and tend to 0 as the index tends to infinity. The default function is that used by Holst, 1987.
- **nstart**: number of starting values: Starting values for the algorithm are formed from the first nstart values of y. The default value is that used in Cameron and Turner, 1993.
- **m0**: value for the initial approximate median; by default, the median of the first nstart observations.
- **scon**: value for the scale parameter s, a function or NULL. When NULL, as by default, the scale is initialized to the mean of the absolute differences between the first nstart y values and m0. If scon is a function, the initial scale is set to scon(y0, m0), where y0 is the vector of the first nstart y values. Note that scon also determines the default for updateScale.
- **updateScale**: a logical indicating if the scale, initialized from scon should be updated in each iteration. Otherwise, the the scale is held constant throughout and the algorithm becomes equivalent to the algorithm of Holst.

Value

An S3 “object” of class "reclas"; simply a list with entries

- **locn**: the successive recursive estimates of location. The first nstart - 1 of these are NA.
- **scale**: the successive recursive estimates of scale if updateScale is true; otherwise the constant value used for the scale.
- **updateScale**: the same as the function argument.
- **call**: the function call, i.e., `match.call`.

There is a plot method for "reclas", see the examples.

Author(s)

<r.turner@auckland.ac.nz> [http://www.stat.auckland.ac.nz/~rolf](http://www.stat.auckland.ac.nz/~rolf)

Extensions by Martin Maechler (scon as function; updateScale, plot()).

References

Examples

```r
set.seed(42)
y <- rt(10000, df = 1.5) # not quite Gaussian ...
z1 <- reclas(y)
z3 <- reclas(y, scon= 1 ) # correct fixed scale
z4 <- reclas(y, scon= 100) # wrong fixed scale
z2 <- reclas(y, # a more robust initial scale:
             scon = function(y0, m0) robustbase::Qn(y0 - m0),
             updateScale = TRUE) # still updated

## Visualizing -- using the plot() method for "reclas":
M <- median(y) ; yl <- c(-1,1)* 0.5
OP <- par(mfrow=c(2,2), mar=.1+c(3,3,1,1), mgp=c(1.5, .6, 0))
plot(z1, M=M, ylim=yl)
plot(z2, M=M, ylim=yl)
plot(z3, M=M, ylim=yl)
plot(z4, M=M, ylim=yl)
par(OP)
```
Index

* L1 Median
  L1median, 7
* Spatial Median
  L1median, 7
* array
  Qrot, 12
* distribution
  rbwheel, 13
* multivariate
  covNNC, 6
  L1median, 7
  mvBACON, 9
* package
  robustX-package, 2
* regression
  BACON, 3
* robust
  BACON, 3
  covNNC, 6
  L1median, 7
  mvBACON, 9
  rbwheel, 13
  reclas, 15
* univar
  reclas, 15
  .lmBACON (BACON), 3
  %\%\%, 13

BACON, 3, 11

class, 16
cov.mcd, 7
cov.nnve (covNNC), 6
covMcd, 7, 9, 11
covNNC, 6
covOGK, 7

function, 16
L1median, 7

l1median_NLM, 9
list, 4, 11

match.call, 16
matrix, 8
median, 9, 16
mvBACON, 3–5, 9

nlm, 8
nlminb, 8
nlminbMethods (L1median), 7

optim, 8
optimMethods (L1median), 7

plot, 16
plot.reclas (reclas), 15

qr, 12, 13
Qrot, 12, 14

rbwheel, 13
reclas, 15
robustX (robustX-package), 2
robustX-package, 2

scale, 14
set.seed, 10