

# Package ‘robustX’

January 2, 2012

**Type** Package

**Title** eXperimental eXtraneous eXtraordinary ... Functionality for Robust Statistics

**Version** 1.1-2

**Date** 2009-12-07

**Author** Werner Stahel, Martin Maechler and potentially others

**Maintainer** Martin Maechler <maechler@stat.math.ethz.ch>

**Description** eXperimental eXtraneous eXtraordinary Functionality for Robust Statistics

**Depends** robustbase

**License** GPL (>= 2)

**Repository** CRAN

**Date/Publication** 2009-12-07 10:54:09

## R topics documented:

robustX-package	2
BACON	3
L1median	5
mvBACON	6
Qrot	8
rbwheel	9

<b>Index</b>	<b>12</b>
--------------	-----------

## Description

The package **robustX** aims to be a collection of 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  
Date: 2008-12-17, was the first, R-forge only  
Version: 1.0-0  
Depends: robustbase  
License: GPL (>= 2)

### Index:

Qrot                    Rotation Matrix to Specific Direction  
rbwheel                Multivariate Barrow Wheel Distribution  
                          Random Vectors  
L1median               Compute the Multivariate L1-Median  
BACON                  BACON for Regression or Multivariate Covariance Estimation  
mvBACON                BACON: Blocked Adaptive  
Computationally-Efficient Outlier Nominators

## Author(s)

Werner Stahel, Martin Maechler and potentially others

Maintainer: Martin Maechler

## See Also

Package **robustbase** which it complements and on which it depends; further package **robust** and the whole CRAN task view on robust statistics, <http://cran.CH.r-project.org/web/views/Robust.html>

## Examples

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

**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**

```
BACON(x, y = NULL, intercept = TRUE,
      m = min(collect * p, n * 0.5),
      init.sel = c("Mahalanobis", "dUniMedian", "random", "manual"),
      man.sel, init.fraction = 0, collect = 4,
      alpha = 0.95, maxsteps = 100, verbose = TRUE)
```

```
## *Auxiliary* function:
```

```
.lmBACON(x, y, intercept = TRUE,
         init.dis, init.fraction = 0, collect = 4,
         alpha = 0.95, maxsteps = 100, verbose = TRUE)
```

**Arguments**

<code>x</code>	a multivariate matrix of dimension [n x p] considered as containing no missing values.
<code>y</code>	the response (n vector) in the case of regression, or NULL for the multivariate case.
<code>intercept</code>	logical indicating if an intercept has to be used for the regression.
<code>m</code>	integer in 1:n specifying the size of the initial basic subset; used only when <code>init.sel</code> is not "manual"; see <code>mvBACON</code> .
<code>init.sel</code>	character string, specifying the initial selection mode; see <code>mvBACON</code> .
<code>man.sel</code>	only when <code>init.sel == "manual"</code> , the indices of observations determining the initial basic subset (and <code>m &lt;- length(man.sel)</code> ).
<code>init.dis</code>	the distances of the x matrix used for the initial subset determined by <code>mvBACON</code> .
<code>init.fraction</code>	if this parameter is > 0 then the tedious steps of selecting the initial subset are skipped and an initial subset of size <code>n * init.fraction</code> is chosen (with smallest dis)
<code>collect</code>	numeric factor chosen by the user to define the size of the initial subset ( <code>p * collect</code> )
<code>alpha</code>	significance level.
<code>maxsteps</code>	the maximal number of iteration steps (to prevent infinite loops)
<code>verbose</code>	logical indicating if messages are printed which trace progress of the algorithm.

## Details

init.sel: the initial selection mode; implemented modes are: "Mah" -> based on Mahalanobis distance (default) "dis" -> based on the distances from the medians "ran" -> based on a random selection "man" -> based on manual selection in this case the vector 'man.sel' which contains the indices of the selected observations must be given. "Mah" and "dis" are proposed by Hadi while "ran" and "man" were implemented in order to study the behaviour of BACON.

## Value

basically a `list` with components

subset	the observation indices (in 1 : n) denoting the subset of “good” observations.
tis	.....

## 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 R, testing etc, by Martin Maechler.

## References

Billor, N., Hadi, A. S., and Velleman, P. F. (2000). BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators; *Computational Statistics and Data Analysis* **34**, 279–298.

## 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)
  (B.ST <- with(starsCYG, BACON(x = log.Te, y = log.light)))
  (RlmST <- lmrob(log.light ~ log.Te, data = starsCYG))
  abline(lmST, col = "red")
  abline(RlmST, col = "blue")
```

L1median

*Compute the Multivariate L1-Median***Description**

Compute the multivariate  $L_1$ -median  $m$ , 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 the data points, the minimization is not entirely trivial.

**Usage**

```
L1median(X, m.init = apply(X, 2, median), 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**

<code>X</code>	numeric <i>matrix</i> of dimension $n \times p$ , say.
<code>m.init</code>	starting value for $m$ ; typically and by default the coordinatewise median.
<code>weights</code>	optional numeric vector of non-negative weights; currently only implemented for method "VardiZhang".
<code>method</code>	character string specifying the computational method, i.e., the algorithm to be used (can be abbreviated).
<code>pscale</code>	numeric $p$ -vector of positive numbers, the coordinate-wise scale (typical size of $\delta m_j$ ), where $m$ is the problem's solution.
<code>tol</code>	positive number specifying the (relative) convergence tolerance.
<code>maxit</code>	positive integer specifying the maximal number of iterations (before the iterations are stopped prematurely if necessary).
<code>trace</code>	an integer specifying the tracing level of the iterations; 0 does no tracing
<code>zero.tol</code>	for method "VardiZhang", a small positive number specifying the tolerance for determining that the iteration is 'exactly' at a data point (which is singularity).
<code>...</code>	optional arguments to <code>nlm()</code> or the control (list) arguments of <code>optim()</code> , or <code>nlminb()</code> , 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’ function, implementing Hossjer and Croux (1995).

**References**

Hossjer and Croux, C. (1995). Generalizing Univariate Signed Rank Statistics for Testing and Estimating a Multivariate Location Parameter. *Non-parametric Statistics* **4**, 293–308.

Vardi, Y. and Zhang, C.-H. (2000). The multivariate  $L_1$ -median and associated data depth. *Proc. National Academy of Science* **97**(4), 1423–1426.

**See Also**

[median](#), [covMcd](#)

**Examples**

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

---

mvBACON

*BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators*

---

**Description**

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

**Usage**

```
mvBACON(x, collect = 4, m = min(collect * p, n * 0.5), alpha = 0.95,
        init.sel = c("Mahalanobis", "dUniMedian", "random", "manual"),
        man.sel, maxsteps = 100, allowSingular = FALSE, verbose = TRUE)
```

**Arguments**

<code>x</code>	numeric matrix (of dimension $[n \times p]$ ), not supposed to contain missing values.
<code>collect</code>	a multiplication factor, when <code>init.sel</code> is not "manual", to define $m$ , the size of the initial basic subset, as $m \leftarrow \min(p * \text{collect}, n/2)$ .
<code>m</code>	integer in $1:n$ specifying the size of the initial basic subset; used only when <code>init.sel</code> is not "manual".
<code>alpha</code>	significance level for the $\chi^2$ cutoff, used to define the next iterations basic subset.
<code>init.sel</code>	character string, specifying the initial selection mode; implemented modes are: <b>"Mahalanobis"</b> based on Mahalanobis distances (default) <b>"dUniMedian"</b> based on the distances from the <b>univariate</b> medians <b>"random"</b> based on a random selection <b>"manual"</b> based on manual selection; in this case, a vector <code>man.sel</code> containing the indices of the selected observations must be specified. "Mahalanobis", "dUniMedian" 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.
<code>man.sel</code>	only when <code>init.sel == "manual"</code> , the indices of observations determining the initial basic subset (and $m \leftarrow \text{length}(\text{man.sel})$ ).
<code>maxsteps</code>	maximal number of iteration steps.
<code>allowSingular</code>	logical indicating a solution should be sought also when no matrix of rank $p$ is found.
<code>verbose</code>	logical indicating if messages are printed which trace progress of the algorithm.

**Value**

	a list with components
<code>subset</code>	logical vector of length $n$ where the $i$ -th entry is true iff the $i$ -th observation is part of the final selection.
<code>dis</code>	numeric vector of length $n$ with the (Mahalanobis) distances.
<code>cov</code>	$p \times p$ matrix, the corresponding robust estimate of covariance.

**Author(s)**

Ueli Oetliker, Swiss Federal Statistical Office, for S-plus 5.1. Port to R, testing etc, by Martin Maechler

**References**

Billor, N., Hadi, A. S., and Velleman, P. F. (2000). BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators; *Computational Statistics and Data Analysis* **34**, 279–298.

**See Also**

[covMcd](#) for a high-breakdown (but more computer intensive) method; [BACON](#) for a “generalization”, notably to *regression*.

**Examples**

```
## 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)
## finds the clear outliers (and 3 "borderline")

## 'coleman' from pkg 'robustbase'
coleman.x <- data.matrix(coleman[, 1:6])
Cc <- covMcd (coleman.x) # truly robust
Cb1 <- mvBACON(coleman.x) ##-> subset is all TRUE hmm??
Cb2 <- mvBACON(coleman.x, init.sel = "dUniMedian")
## --> BACON "breaks down" here
```

---

Qrot

*Rotation Matrix to Specific Direction*


---

**Description**

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

**Usage**

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

**Arguments**

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

**Details**

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

**Value**

$p \times p$  orthogonal matrix which rotates  $(1, 0, \dots, 0)$  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-triagonal 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**

<code>n</code>	integer, specifying the sample size.
<code>p</code>	integer, specifying the dimension (aka number of variables).
<code>frac</code>	numeric, the proportion of outliers. The default, $1/p$ , corresponds to the (asymptotic) breakdown point of M-estimators.
<code>sig1</code>	thickness of the "wheel", ( $= \sigma(\text{good}[, 1])$ ), a non-negative numeric.
<code>sig2</code>	thickness of the "axis" (compared to 1).
<code>rGood</code>	function; the generator for "good" observations.
<code>rOut</code>	function, generating the outlier observations.
<code>U1</code>	$p$ -vector to which $(1, 0, \dots, 0)$ is rotated.
<code>scaleAfter</code>	logical indicating if the matrix is re-scaled <i>after</i> rotation (via <code>scale()</code> ). Default TRUE; note that this used to be false by default in the first public version.
<code>scaleBefore</code>	logical indicating if the matrix is re-scaled before rotation (via <code>scale()</code> ).
<code>spherize</code>	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).
<code>fullResult</code>	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,  $n1 \approx (1 - f) \cdot n$ ,  $f = \text{frac}$ .

If `fullResult` is `TRUE`, a list with components

<code>X</code>	the $n \times p$ matrix of above, $X = X0 \%*\% A$ , where $A \leftarrow \text{Qrot}(p, u = U1)$ , and $X0$ is the corresponding matrix before rotation, see below.
<code>X0</code>	.....
<code>A</code>	the $p \times p$ rotation matrix, see above.
<code>n1</code>	the number of “good” observations, see above.
<code>n2</code>	the number of “outlying” observations, $n2 = n - n1$ .

**Author(s)**

Werner Stahel and Martin Maechler

**References**

<http://stat.ethz.ch/research/areas/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")) {
  # 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)
```

# Index

- \*Topic **array**
  - Qrot, 8
- \*Topic **distribution**
  - rbwheel, 9
- \*Topic **multivariate**
  - L1median, 5
  - mvBACON, 6
- \*Topic **package**
  - robustX-package, 2
- \*Topic **regression**
  - BACON, 3
- \*Topic **robust**
  - BACON, 3
  - L1median, 5
  - mvBACON, 6
  - rbwheel, 9
- .lmBACON (BACON), 3
- %%%, 8
  
- BACON, 3, 7
  
- covMcd, 6, 7
  
- L1median, 5
- list, 4
  
- matrix, 5
- median, 6
- mvBACON, 3, 4, 6
  
- nlm, 5
- nlminb, 5
- nlminbMethods (L1median), 5
  
- optim, 5
- optimMethods (L1median), 5
  
- qr, 8
- Qrot, 8, 10
  
- rbwheel, 9
  
- robustX (robustX-package), 2
- robustX-package, 2
  
- scale, 9