Package ‘robustX’

February 25, 2019

Type Package
Title 'eXtra' / 'eXperimental' Functionality for Robust Statistics
Version 1.2-4
Date 2019-02-25
Author Werner Stahel,
       Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>)
       and potentially others
Maintainer Martin Maechler <maechler@stat.math.ethz.ch>
Description Robustness -- 'eXperimental', 'eXtraneous', or 'eXtraordinary'
       Functionality for Robust Statistics. In other words, methods which are not
       yet well established, often related to methods in package 'robustbase'.
Imports grDevices, graphics, stats, utils, robustbase (>= 0.92-3)
Suggests MASS, lattice
Enhances ICS
License GPL (>= 2)
Encoding UTF-8
Repository CRAN
Repository/R-Forge/Project robustbase
Repository/R-Forge/Revision 836
Repository/R-Forge/DateTimeStamp 2019-02-25 09:25:22
Date/Publication 2019-02-25 11:30:03 UTC
NeedsCompilation no

R topics documented:

robustX-package .......................... 2
BACON .................................. 3
covNNC .................................. 5
L1median ................................. 7
mvBACON ................................. 9
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
Type: Package
Title: 'eXtra' / 'eXperimental' Functionality for Robust Statistics
Version: 1.2-4
Date: 2019-02-25
Author: Werner Stahel, Martin Maechler [aut, cre] (<https://orcid.org/0000-0002-8685-9910>) and potentially others
Maintainer: Martin Maechler <maechler@stat.math.ethz.ch>
Description: Robustness – 'eXperimental', 'eXtraneous', or 'eXtraordinary' Functionality for Robust Statistics. In other words, methods which are not yet well established, often related to methods in package 'robustbase'.
Imports: grDevices, graphics, stats, utils, robustbase (>= 0.92-3)
Suggests: MASS, lattice
Enhances: ICS
License: GPL (>= 2)
Encoding: UTF-8
Repository: R-Forge
Repository/R-Forge/Project: robustbase
Repository/R-Forge/Revision: 836
Repository/R-Forge/DateTimeStamp: 2019-02-25 09:25:22
Date/Publication: 2019-02-25 09:25:22

Index of help topics:

- Qrot
- rbwheel
- reclas

- BACON
  BACON for Regression or Multivariate Covariance Estimation

- L1median
  Compute the Multivariate L1-Median aka 'Spatial Median'

- Qrot
  Rotation Matrix to Specific Direction

- covNNC
  Robust Covariance Estimation via Nearest Neighbor Cleaning

- mvBACON
  BACON: Blocked Adaptive
BACON

Computationally-Efficient Outlier Nominators

rbwheel
Multivivariate Barrow Wheel Distribution Random Vectors

reclas
Recursive Robust Median-like Location and Scale

robustX-package
Eperimental extraneous ... Functionality for Robust Statistics

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.r-project.org/web/views/Robust.html

Examples

pairs( rbwheel(100, 4) )

BACON

BACON for Regression or Multivariate Covariance Estimation

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 .1mBACON() 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:
.hlBACON(x, y, intercept = TRUE,
  init.dis, init.fraction = 0, collect = 4,
  alpha = 0.95, maxsteps = 100, verbose = TRUE)
Arguments

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

Details

Notably about the initial selection mode, **init.sel**, see its description in the **mvBACON** arguments list.

Value

**BACON(x,y,...)** (for regression) returns a **list** with components

- **subset**: the observation indices (in 1: \(n\)) denoting a subset of “good” supposedly outlier-free observations.
- **tis**: the \(t_i(y_n, X_m)\) 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.
covNNC

Description


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. 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).

References


See Also

mvBACON, the multivariate version of the BACON algorithm.

Examples

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

```r
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`
- `innnc`: 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.dlk()`). The original version is available, unexported as `robustX::covNNC1`. 
References


See Also

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

The whole package rrcov.

Examples

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 = colMedians(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; \( \emptyset \) 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 \texttt{nlm()} or the control (list) arguments of \texttt{optim()}, or \texttt{nlminb()}, respectively.

Details

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

Value

- currently the result \textit{depends} strongly on the method used.
- FIXME. This will change considerably.

Author(s)

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

References

mvBACON

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 Nomi-

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

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

x numeric matrix (of dimension \([n \times p]\)), not supposed to contain missing values.
collect a multiplication factor \(c\), when init.sel is not "manual", to define \(m\), the size of the initial basic subset, as \(m := c \cdot p\). In practice, \(m \leftarrow \min(p \times \text{collect}, n/2)\).
m integer in \([1:n]\) specifying the size of the initial basic subset; used only when init.sel is not "manual".
alpha significance level for the \(\chi^2\) cutoff, used to define the next iterations basic subset.
init.sel character string, specifying the initial selection mode; implemented modes are:
"Mahalanobis" based on Mahalanobis distances (default); the version V1 of the reference; affine invariant but not robust.
mvBACON

"dUniMedian" based on the distances from the univariate medians; the version V2 of the reference; robust but not affine invariant.

"random" based on a random selection, i.e., reproducible only via set.seed().

"manual" based on manual selection; in this case, a vector man.sel 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.

man.sel only when init.sel == "manual", the indices of observations determining the initial basic subset (and n <- 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.

Value

a list with components

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 R, testing etc, by Martin Maechler

References


See Also

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

Examples

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,9L,11L,14L,28L,30L,34L)))
qrot

## Rotation Matrix to Specific Direction

### Description

Construct the \( p \times p \) rotation matrix that rotates the unit vector \((1, 0, \ldots, 0)\), i.e., the \( x_1 \)-axis, onto \((1, 1, 1, \ldots, 1)/\sqrt{p}\), 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 \( \text{“the diagonal”} \propto (1, 1, 1, \ldots, 1) \).

### Details

The \( \text{qr} \) decomposition is used for a Gram-Schmitt basis orthogonalization.

### Value

\( p \times p \) orthogonal matrix which rotates \((1, 0, \ldots, 0)\) onto a vector proportional to \(\text{unit.image}\).

### Author(s)

Martin Maechler
rbwheel

See Also

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

Examples

```r
Q <- qrrot(6)
zapsmall(crossprod(Q)) # 6 x 6 unity \leftrightarrow Q'Q = I \leftrightarrow Q orthogonal

if(require("MASS")) {
  Qt <- qrrot6(transpose = TRUE)
  stopifnot(all.equal(Qt, t(Q)))
  fractions(Qt ^ 2) # \rightarrow 1/6 1/30 etc, in an almost lower-triaggonal matrix
}
```

rbwheel

Multivariate Barrow Wheel Distribution Random Vectors

Description

Generate \(p\)-dimensional random vectors according to Stahel’s Barrow Wheel Distribution.

Usage

```r
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, sph erize = 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, $n1 \approx (1 - f) \cdot n$, $f = \frac{\cdot}{\cdot}$.

If fullResult is TRUE, a list with components

- $X$ the $n \times p$ matrix of above, $X = X0 \%*% A$, where $A < Qrot(p, u = U1)$, and $X0$ 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, $n2 = n - n1$.

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
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

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

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 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 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

*Topic array
  Qrot, 11

*Topic distribution
  rbwheel, 12

*Topic multivariate
  covNNC, 5
  L1median, 7
  mvBACON, 9

*Topic package
  robustX-package, 2

*Topic regression
  BACON, 3

*Topic robust
  BACON, 3
  covNNC, 5
  L1median, 7
  mvBACON, 9
  rbwheel, 12
  reclas, 14

*Topic univar
  reclas, 14
  .lmBACON (BACON), 3
  %**, 12

BACON, 3, 10

class, 15

cov.mcd, 7

cov.nnve (covNNC), 5

covMcd, 7, 9, 10

covNNC, 5

covOGK, 7

function, 15

L1median, 7

l1median_NLM, 9

list, 4

match.call, 15

matrix, 8

median, 9, 15

mvBACON, 3–5, 9

nlm, 8

nlminb, 8

nlminbMethods (L1median), 7

optim, 8

optimMethods (L1median), 7

plot, 15

plot.reclas (reclas), 14

qr, 11, 12

Qrot, 11, 13

rbwheel, 12

reclas, 14

robustX (robustX-package), 2

robustX-package, 2

scale, 12

set.seed, 10