Package: OjaNP (via r-universe)

October 13, 2026

Type Package

Title Multivariate Methods Based on the Oja Median and Related Concepts

Version 2.0

Date 2026-06-26

Encoding UTF-8

Depends R (>= 4.0), ICS, ICSNP

LinkingTo Rcpp

Description Calculating the Oja median, Oja signs and ranks and methods based upon them. For details, see the the corresponding publication Fischer, Mosler, et al. (2020) <doi:10.18637/jss.v092.i08>.

License GPL (>= 2)

Suggests testthat (>= 3.0.0)

LazyLoad yes

NeedsCompilation yes

Config/testthat/edition 3

Author Daniel Fischer [aut, cre] (ORCID: <https://orcid.org/0000-0002-7513-683X>), Juho Eagling [aut] (ORCID: <https://orcid.org/0000-0001-8333-5057>), Karl Mosler [aut] (ORCID: <https://orcid.org/0000-0002-4015-8094>), Jyrki Möttönen [aut] (ORCID: <https://orcid.org/0000-0002-6270-2556>), Klaus Nordhausen [aut] (ORCID: <https://orcid.org/0000-0002-3758-8501>), Oleksii Pokotylo [aut] (ORCID: <https://orcid.org/0000-0002-1136-5123>), Daniel Vogel [aut], Tommi Ronkainen [ctb] (Contributor of incorporated C++ code), Makoto Matsumoto [ctb, cph] (Original Mersenne Twister implementation), Takuji Nishimura [ctb, cph] (Original Mersenne Twister implementation), Shawn Cokus [ctb] (Contributor to the Mersenne Twister implementation), Richard J. Wagner [ctb] (Contributor to the Mersenne Twister implementation)

Maintainer Daniel Fischer <daniel.fischer@luke.fi>

Repository https://cran.r-universe.dev

Date/Publication 2026-07-21 11:20:09 UTC

RemoteUrl https://github.com/cran/OjaNP

RemoteRef HEAD

RemoteSha 2509a397e09d73650f6b4e8e83871156aaf03b76

Contents

OjaNP-package 2
biochem 3
hyperplane 4
oja1sampleTest 5
ojaCsampleTest 7
ojaMedian 9
ojaMedianControl 11
ojaMedianFn 13
ojaRank 14
ojaRCM 16
ojaSCM 19
ojaSign 21
ojaSignedRank 24

Index **27**

`OjaNP-package` *Multivariate Methods Based on the Oja Median and Related Concepts*

Description

The package provides functions for the Oja median, Oja signs and ranks and methods based upon them.

Details

Package:	OjaNP
Type:	Package
Version:	2.0
Date:	2026-06-26
License:	GPL (>= 2)
LazyLoad:	yes

The package implements several algorithms to compute the Oja median as well as the Oja signs and ranks and methods based upon them.

Author(s)

Daniel Fischer, Juho Eagling, Karl Mosler, Jyrki Möttönen, Klaus Nordhausen, Oleksii Pokotylo, Daniel Vogel.

Maintainer: Daniel Fischer <daniel.fischer@luke.fi>

`biochem` *Biochemical Data*

Description

The data consists of levels of biochemical components in brains of mice. The treatment group received a drug and the control group a placebo.

Usage

```
data(biochem)
```

Format

A data frame with 22 observations on the following 3 variables.

`comp.1` First biochemical component

`comp.2` Second biochemical component

`group` Factor with the two levels 'Control' and 'Treat'

Source

Table 1 in Brown and Hettmansperger (1987).

References

Brown, B. M. and Hettmansperger, T. P. (1987). Affine invariant rank methods in the bivariate location model. Journal of the Royal Statistical Society, Series B, ***49***, *301–310.*

Examples

```
data(biochem)
ojaMedian(biochem[,1:2])
```

`hyperplane` *Hyperplane Passing Through k Points in the k-dimensional Space.*

Description

The function computes a $(k-1)$-dimensional hyperplane passing through k given points in the k-dimensional space.

Usage

```
hyperplane(X)
```

Arguments

`X`	a numeric $k \times k$ matrix containing k data points as **rows**.

Details

A $(k-1)$-dimensional hyperplane in R^k consists of all points x that satisfy

$$d^T x + c = 0,$$

where d is a k-vector and c is a scalar. The function returns the $(k+1)$-vector (d, c). It is normalized such that the length of d equals $(k-1)!$ times the $(k-1)$-dimensional volume of the simplex formed by the points on the plane. (If $k = 3$, this is a triangle.) Hence this function can also easily be used to compute volumes of simplices.

The direction of d, that is, whether it points towards the origin or not, is not fixed. It depends on the order of the data points within the matrix `X`.

If the k points do not uniquely define a $(k-1)$-dimensional hyperplane (i.e. they lie on a $(k-2)$-dimensional hyperplane), a vector containing zeros is returned.

Value

a vector of length $(k+1)$ describing the hyperplane, see details above.

Author(s)

Daniel Vogel

Examples

```
### ----<< Example 1 >>---- : line in R^2
X <- rbind(c(4,5),c(8,2))
hyperplane(X)
# The line through the the points c(4,5) and c(8,2) is given by
#    3*x + 4*y - 32 = 0.
# The norm of the first two components of the return value
# of hyperplane() (i.e. the vector d above) equals the
```

```
# distance of both points.

X <- rbind(c(8,2),c(4,5))
hyperplane(X)
# If the order of the points is changed, the direction of d
# (see details) may also change.



### ----<< Example 2 >>---- : unit vectors in R^3
X <- diag(rep(1,3))
hyperplane(X)
# The plane passing through all three unit vectors is given by
#   -x - y - z + 1 = 0.
# These three points form a equilateral triangle on the plane
# with side length sqrt(2) and hence area sqrt(3)/2.
# The norm of d (see details) equals twice this number.
```

`oja1sampleTest` *One Sample Location Test Based on Oja Signs and Ranks*

Description

Function to test for location in the one sample case using Oja signs and ranks.

Usage

```
oja1sampleTest(X, mu = NULL, scores = "sign", p = 1,
               method = "approximation", n.simu = 1000,
               na.action = na.fail, ...)
```

Arguments

`X`	a numeric data frame or data matrix.
`mu`	a vector indicating the hypothesized value of the location. NULL represents the origin.
`scores`	options are "rank" for the signed rank test, "sign" for the sign test. The sign test is the default.
`p`	value of "p" to be passed on to `ojaSign` or `ojaSignedRank`. The default here is to use all hyperplanes since only then the tests are valid. This can make the functions quite slow, however.
`method`	defines the method used for the computation of the p-value. The possibilities are "approximation" (default) or "permutation" which permutes the signs of the Oja signs or Oja signed ranks.
`n.simu`	if "`method` = permutation" specifies this the number of replications used in the permutation procedure.
`na.action`	a function which indicates what should happen when the data contain 'NA's. Default is to fail.
`...`	further arguments to be passed to or from methods.

Value

A list with class 'htest' containing the following components:

`statistic`	the value of the Q-statistic.
`parameter`	the degrees of freedom for the Q-statistic or the number of replications in the permutation procedure.
`p.value`	the p-value for the test.
`null.value`	the specified hypothesized value of the location.
`alternative`	a character string with the value 'two.sided'.
`method`	a character string indicating what type of test was performed.
`data.name`	a character string giving the name of the data.

Author(s)

Klaus Nordhausen

References

Hettmansperger, T. P., Nyblom, J. and Oja, H. (1994), Affine invariant multivariate one-sample sign test, Journal of the Royal Statistical Society, *Series B*, ***56***, *221–234.*

Hettmansperger, T. P., Möttönen, J. and Oja, H. (1997), Multivariate affine invariant one-sample signed-rank tests, Journal of the American Statistical Society, ***92***, *1591–1600.*

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92****(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

See Also

`ojaMedian`, `ojaSign`, `ojaSignedRank`

Examples

```
data(biochem)
oja1sampleTest(biochem[,1:2], mu = c(1.1, 0.4))
oja1sampleTest(biochem[,1:2], mu = c(1.1, 0.4), method = "p")

oja1sampleTest(biochem[,1:2], mu = c(1.1, 0.4), scores = "rank")
oja1sampleTest(biochem[,1:2], mu = c(1.1, 0.4), scores = "rank", method = "p")
```

ojaCsampleTest *C Sample Location Test Based on Oja Signs and Ranks*

Description

Function to test for equality of location in the C sample case using Oja signs and ranks.

Usage

```
ojaCsampleTest(X, ...)

## Default S3 method:
ojaCsampleTest(X, Y, mu = NULL, scores = "sign", p = 1,
               method = "approximation", n.simu = 1000,
               center = "ojaMedian", na.action = na.fail, ...)

## S3 method for class 'formula'
ojaCsampleTest(formula, scores="sign", p = 1,
               method = "approximation", n.simu = 1000,
               center = "ojaMedian", data, subset, na.action,...)
```

Arguments

X	a numeric data frame or matrix in the two sample case.
Y	a numeric data frame or matrix in the two sample case.
formula	a formula of the form X ~ g where X is a numeric matrix with at least two columns giving the data values and g a factor with at least two levels giving the corresponding groups.
mu	a vector indicating the hypothesized value of the difference in location. NULL represents no difference between the groups. For more than two groups mu should be 0 or not be specified at all.
scores	options are "rank" for the Oja rank test, "sign" for the Oja sign test. The sign test is the default.
p	value of "p" to be passed on to ojaSign or ojaRank. The default here is to use all hyperplanes since only then the tests are valid. This can make the functions quite slow, however.
method	defines the method used for the computation of the p-value. The possibilities are "approximation" (default) or "permutation".
n.simu	if "method = permutation" specifies this the number of replications used in the permutation procedure.
center	value of "center" to be passed on to ojaSign. Is used to center the data matrix. The default is the natural but computationally expansive Oja median. For other options see the help for ojaSign.

data	an optional data frame, list or environment containing the variables in the model. If not found in "data", the variables are taken from "environment(formula)".
subset	an optional vector specifying a subset of observations to be used for the testing.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.
...	further arguments to be passed to or from methods. Mainly further arguments for ojaSign or ojaRank.

Details

In the C-sample case of the Oja sign test the covariance matrix of the signs is divided by the sample size and not by sample size - 1.

For the sign test version always the Oja median should be used from a theoretical point of view to center the data and the median should be computed using the exact algorithm. For further details about the the Oja median see ojaMedian.

Note that no theoretical results are available when "p" is not set to 1.

Value

A list with class 'htest' containing the following components:

statistic	the value of the Q-statistic.
parameter	the degrees of freedom for the Q-statistic or the number of replications in the permutation procedure.
p.value	the p-value for the test.
null.value	the specified hypothesized value of the difference in location. (only in the two sample case)
alternative	a character string with the value 'two.sided'. (only in the two sample case)
method	a character string indicating what type of test was performed.
data.name	a character string giving the name of the data.

Author(s)

Klaus Nordhausen

References

Hettmansperger, T. P. and Oja, H. (1994), Affine invariant multivariate multisample sign test, Journal of the Royal Statistical Society, *Series B*, ***56***, *235–249.*

Hettmansperger, T. P., Möttönen, J. and Oja, H. (1999), Multivariate affine invariant rank tests for several samples, Statistica Sinica, ***8***, *785–800.*

Visuri, S., Ollila, E., Koivunen, V., Möttönen, J. and Oja, H. (2003), Affine equivariant multivariat rank methods, Journal of Statistical Planning and Inference, ***114***, *161–185.*

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92****(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

See Also

ojaSign, ojaRank, oja1sampleTest

Examples

```
data(biochem)
X <- subset(biochem, group=="Control", select=c("comp.1","comp.2"))
Y <- subset(biochem, group=="Treat", select=c("comp.1","comp.2"))
ojaCsampleTest(X,Y, alg="exact")
ojaCsampleTest(X,Y, method="p", alg="exact")
ojaCsampleTest(cbind(comp.1, comp.2) ~ group, score="r", data=biochem)
```

---

ojaMedian *Oja Median*

Description

Function to compute the Oja median. Several algorithms are possible.

Usage

```
ojaMedian(X, alg = "evolutionary", sp = 1, na.action = na.fail,
          control = ojaMedianControl(...), ...)

ojaMedianEvo(X, control = ojaMedianControl(...), ...)
ojaMedianGrid(X, control = ojaMedianControl(...), ...)
ojaMedianEx(X, control = ojaMedianControl(...), ...)
ojaMedianExB(X, control = ojaMedianControl(...), ...)
```

Arguments

X	numeric data.frame or matrix.
alg	character string denoting the algorithm to be used for computing the Oja median. Options are "exact", "bounded_exact", "evolutionary" and "grid". Default is "evolutionary". See Details.
sp	number of runs to average over.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.
control	a list specifying the control parameters of the different algorithms; use the function ojaMedianControl and see its help page.
...	can be used to specify control parameters directly instead of via control.

Details

There are four possible algorithms to calculate the Oja median. The exact algorithm uses a gradient method. It follows intersection lines of hyperplanes until it reaches the minimum of an objective function. It is computationally a very intensive algorithm and it calculates the Oja median in acceptable time in the bivariate case for at least 1200 datapoints. For a 7-dimensional dataset it is possible to calculate it for 24 datapoints.

The bounded exact algorithm modifies the exact algorithm by employing bounded regions which contain the median. The regions are built using the centered rank function. The new algorithm is faster and has less complexity. Parameter `volume` is the desired size of the bounded region, which is selected as a part of the original volume. Here the volume is calculated as the volume of a minimal multivariate circumscribed rectangle with edges parallel to the coordinate axes. Setting parameter `boundedExact` to `FALSE` stops the algorithm after the bounded region is found, and its center is reported as an approximation of the median.

With the evolutionary algorithm it is possible to calculate an approximative solution. It starts with a random point and mutates this temporary best solution in order to gain a better one. There are several options to control the mutation process. If you are interested in a fast calculation of the Oja median and you tolerate a higher error rate, you should set `sigmaAdaption` to 1. As a second possibility you could limit the number of subsets used to a small number. If you use all subsets, there are in total n choose k, with n number of datapoints and k dimensions. If you are interested in a precise solution, the following options have turned out to be useful: `initialSigma`: 0.5, `sigmaAdaptation`: 20, `adaptationFactor`: 0.5, `sigmaLog20Decrease`: 10. Tests have been made in the bivariate case, but these values should work for every dimension. In the bivariate case it is possible to calculate the Oja median for more than $22 * 10^6$ datapoints. In the 10-dimensional case the algorithm is still able to calculate an approximative solution for 10^6 datapoints. Before the algorithm starts itself we transform the data with ICS in order to get a more stable version (with respect to the location of the data) and improve the quality of the approximation. Another reason for this was to get an affine invariant way of the approximation.

The fourth algorithm calculates the Oja median by means of a grid. The grid points are possible approximations of the Oja median. Every grid point is tested to be the Oja median. If the test results are not unique the algorithm will take a bigger sample of subsets into account and test it again. In comparison to the evolutionary algorithm it is slower and less precise. Only in special data situations it might be useful. The algorithm constitutes an earlier heuristical solution to the Oja median problem and is included mainly for historical reasons.

The exact algorithm and the grid algorithm are also described in Ronkainen et al. (2002). The bounded search algorithm is described in Mosler and Pokotylo (2015).

A lot of calculation time in the `ojaMedian` function might be spend for checking the input and for transforming it. So if you do time-critical calculations, e.g. with loops, you might want to take the variants `ojaMedianEx`, `ojaMedianExB`, `ojaMedianEvo` or `ojaMedianGrid`. Please use this only if you know what you are doing, because there are no checks, just the `.Call` to the algorithm itself.

If the dimension of your data is too big or if there are too many observations, it is possible that the exact algorithm will crash R. On a common PC with a 32-bit operating system the following combinations of dimension:amount will work fine: 2:1200, 3:300, 4:100, 5:63, 6:38, 7:24. Bigger datasets might be possible, depending on your system.

Another general restriction with this function is that there should be more data points than dimensions.

There is a demo available which demonstrates graphically the Oja median in simple data situations in the bivariate case. To view the demo run `demo(ojaMedianDemo)`.

Value

a numeric vector containing the Oja median.

Author(s)

Ported to R by Daniel Fischer. Original C++-code by Thorsten Bernholt, Robin Nunkesser and Tommi Ronkainen. Bounded search algorithm by Oleksii Pokotylo.

References

Oja, H. (1983), Descriptive statistics for multivariate distributions, Statistics and Probability Letters, ***1***, *327–332.*

Ronkainen, T., Oja, H. and Orponen, P. (2002), Computation of the multivariate Oja median, in Dutter R., Filzmoser P.,Gather U. and Rousseeuw, P. J.: Developments in Robust Statistics, *Heidelberg: Springer, 344–359.*

Fischer, D. (2008), Diplomarbeit, *Statistische Eigenschaften des Oja-Medians mit einer algorithmischen Betrachtung, Dortmund: Technische Universität Dortmund. In German.*

Mosler, K. and Pokotylo, O. (2015), "Computation of the Oja Median by Bounded Search." Modern Nonparametric, Robust and Multivariate Methods. *Springer International Publishing, 185–203.*

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92****(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

Examples

```
data(biochem)
X <- as.matrix(biochem[,1:2])
ojaMedian(X)
ojaMedian(X, alg = "evo")
ex <-ojaMedian(X, alg = "exact")
exb<-ojaMedian(X, alg = "bounded_exact")

ojaMedianFn(X, ex)
ojaMedianFn(X, exb)
```

`ojaMedianControl` *Tuning Parameters for the Function 'ojaMedian'*

Description

Tuning parameters for the algorithms used by function `ojaMedian`.

Usage

```
ojaMedianControl(sigmaInit = 0, sigmaAda = 20, adaFactor = 0.5,
                 iter = 1e+06, useAllSubsets = FALSE,
                 nSubsetsUsed = 1000, sigmaLog10Dec = 10,
                 storeSubDet = TRUE, eps = 0.1, chi2 = 0.95,
                 samples = 20, maxlines = 1000, S1 = cov,
                 S2 = cov4, S1args = list(), S2args = list(),
                 volume = 1e-6, boundedExact = TRUE, ...)
```

Arguments

`sigmaInit`	(for the `evo` algorithm): Set the initial variance of the mutation vector in the first run.
`sigmaAda`	(for the `evo` algorithm): Defines after how many mutations the variance of the mutation vector is adjusted.
`adaFactor`	(for the `evo` algorithm): Defines the level of adjustment of the mutation vector.
`iter`	(for the `evo` algorithm): The maximum number of iterations. If the algorithm does not converge, it stops after `iter` - iterations.
`useAllSubsets`	(for the `evo` algorithm): A logical flag. If it is set all datapoints and resulting simplices are taken into account for the calculation.
`nSubsetsUsed`	(for the `evo` algorithm): If useAllSubsets is not set, this determines how many, randomly selected, datapoints are taken into account.
`sigmaLog10Dec`	(for the `evo` algorithm): This is an abort criterion. If the logarithmised initial variance differs more than sigmaLog10Dec from the actual, logarithmised variance, the algorithm stops.
`storeSubDet`	(for the `evo` algorithm): A boolean flag. If it is set subdeterminants are stored. This should always been set to `TRUE` if $6*(dim-1)*nSubsetsUsed < CPU-Cache$.
`eps`	(for the `grid` algorithm): This is the abort criterion. If the grid becomes denser than this threshold the algorithm stops.
`chi2`	(for the `grid` algorithm): This is the test niveau of the test, if a grid point could be used as a Oja-Median or not.
`samples`	(for the `grid` algorithm): This determines how many additional hyperplanes are taken after every run.
`maxlines`	(for the `exact` algorithm): This determines how many intersection lines are investigated in addtion to the one with the steepest gradient.
`S1`	(for the `evo` and `grid` algorithms): Passed on to `ics` to compute the invariant coordinate system. Default is `cov`.
`S2`	(for the `evo` and `grid` algorithms): Passed on to `ics` to compute the invariant coordinate system. Default is `cov4`.
`S1args`	(for the `evo` and `grid` algorithms): Optional arguments for `S1` passed on to `ics` to compute the invariant coordinate system.
`S2args`	(for the `evo` and `grid` algorithms): Optional arguments for `S2` passed on to `ics` to compute the invariant coordinate system.

volume	(for the bounded_exact algorithm): is the desired size of the bounded region, which is selected as a part of the original volume. Here the volume is calculated as the volume of a minimal multivariate circumscribed rectangle with edges parallel to the coordinate axes.
boundedExact	(for the bounded_exact algorithm): setting this parameter to FALSE stops the algorithm after the bounded region is found, and its center is reported as an approximation of the median.
...	Additional hidden option

Value

a list with parameters set to calculate the Oja Median with ojaMedian.

Author(s)

Daniel Fischer and Klaus Nordhausen

See Also

ojaMedian, also for references and examples.

Examples

```
## Show the default settings:
str(ojaMedianControl())
```

---

ojaMedianFn *Value of the Oja Criterion*

Description

The function returns the value of the Oja criterion function for a given point.

Usage

```
ojaMedianFn(X, x)
```

Arguments

X	a numeric data set or data matrix.
x	a numeric vector with the coordinates of interest.

Value

a numeric value containing the ciriterion evaluated at x for the data X.

Author(s)

ported to R by Daniel Fischer. Original C++ code by Tommi Ronkainen.

References

Oja, H. (1983), Descriptive statistics for multivariate distributions, Statistics and Probability Letters, ***1***, *327–332.*

Ronkainen, T., Oja, H. and Orponen, P. (2002), Computation of the multivariate Oja median, in Dutter R., Filzmoser P.,Gather U. and Rousseeuw, P. J.: Developments in Robust Statistics, *Heidelberg: Springer, 344–359.*

See Also

ojaMedian

Examples

```
data(biochem)
X <- as.matrix(biochem[,1:2])
x <- ojaMedian(X)
ojaMedianFn(X, x)
ojaMedianFn(X, c(1.1, 0.4))
```

---

ojaRank — *Oja Ranks – Affine Equivariant Multivariate Ranks*

Description

The function computes the Oja rank of a point x w.r.t. a data set X or, if no point x is given, the Oja ranks of all points in X.

Usage

```
ojaRank(X, x = NULL, p = NULL, silent = FALSE, na.action = na.fail)
```

Arguments

X	numeric data.frame or matrix containing the data points as rows.
x	NULL or a numeric vector, the point for which the Oja rank should be computed.
p	NULL or a number between 0 and 1 which specifies the fraction of hyperplanes to be used for subsampling. If p = 1, no subsampling is done. If p = NULL, the value of p is determined based on the size of the data set. See details.
silent	logical, if subsampling is done or the expected computation time is too long, a warning message will be printed unless silent is TRUE. The default is FALSE.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.

Details

The function computes the Oja rank of the point x w.r.t.\ the data set X or, if no x is specified, the Oja ranks of all data points in X w.r.t. X. For a definition of *Oja rank* see reference below.

The matrix X needs to have at least as many rows as columns in order to give sensible results. The vector x has to be of length ncol(X). If x is specified, a vector of length ncol(X) is returned. Otherwise the return value is a matrix of the same dimensions as X where the i-th row contains the Oja rank of the i-th row of X.

The function will also work for matrices X with only one column and also vectors. Then centered and normalized (univariate) ranks are returned.

For $n = nrow(X)$ data points in R^k, where $k = ncol(X)$, the computation of the Oja rank necessitates the evaluation of $N = choose(n, k)$ hyperplanes in R^k. Thus for large data sets the function offers a subsampling option in order to deliver (approximate) results within reasonable time. The subsampling fraction is controlled by the parameter p: If $p < 1$ is passed to the function, the computation is based on a random sample of only pN of all possible N hyperplanes. If p is not specified (which defaults to p = NULL), it is automatically determined based on n and k to yield a sensible trade-off between accuracy and computing time. If $Nk^3 < 6 \cdot 10^6$, the sample fraction p is set to 1 (no subsampling). Otherwise p is choosen such that the computation (of one rank) *usually* takes around 20 seconds (on a 1.66 GHz CPU and 1 GB RAM). If all Oja ranks of X are requested, a hyperplane sample is drawn once, all Oja ranks are then computed based on this sample.

Finally, subsampling is feasible. Even for very small p useable results can be expected, see e.g. the examples for the function ojaRCM.

Claudia Köllmann is acknowledged for bug-fixing this function.

Value

either a numeric vector, the Oja rank of x, or a matrix of the same dimensions as X containing the Oja ranks of X as rows.

Author(s)

Daniel Vogel, Jyrki Möttönen

References

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92***(8), *pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

Oja, H. (1999), Affine invariant multivariate sign and rank tests and corresponding estimates: A review, Scand. J. Statist., ***26***, *319–343.*

See Also

ojaSign, ojaRCM, hyperplane, ojaSignedRank

Examples

```
### ----<< Example 1 >>---- : 30 points in R^2
set.seed(123)
X <- rmvnorm(n = 30,mean = c(0,0)) # from package 'mvtnorm'
ojaRank(X)
ojaRank(X, x = c(100,100))
ojaRank(X, x = ojaMedian(X, alg="exact")) # close to zero

# The following two return the same (only in different time)
ojaRank(X)
t(apply(X, 1, function(y){ojaRank(X,y)}))
# but the following two do not (due to different subsampling).
# 1)
set.seed(123); ojaRank(X, p = 0.9, silent = TRUE)
# 2)
set.seed(123)
t(apply(X, 1, function(y){ojaRank(X, y, p = 0.9, silent = TRUE)}))
# In 1) one subsample for all ranks is drawn, whereas in 2)
# a different sample for each rank is drawn.

### ----<< Example 2 >>---- : three points in R^3: only one hyperplane
# The following commands return the same result.
ojaRank(X = diag(rep(1, 3)), x = c(0,0,0))
ojaRank(X = diag(rep(1, 3)), x = c(-100,-110,-5550))
hyperplane(X = diag(rep(1,3)))[1:3]


### ----<< Example 3 >>---- : 300 points in R^7
# Subsampling is done.
# The following example might take a bit longer:

set.seed(123)
X <- rmvnorm(n = 300, mean = rep(0, 7))
system.time(or <- ojaRank(x = 1:7, X = X))
# PLEASE NOTE: The computation of the Oja rank is based on a
# random sub-sample of less than 1% of all possible hyperplanes.
#
#       user      system     elapsed
#      18.47        0.00       18.47
print(or,d=4)
# [1]  7.733  6.613  6.839  7.383 18.237 21.851 23.700



### ----<< Example 4 >>---- : univariate ranks
ojaRank(1:10)
ojaRank(X = 1:10, x = 5.5)
```

---

ojaRCM *Oja Rank Covariance Matrix*

Description

The function computes the Oja rank covariance matrix of a data set X.

Usage

```
ojaRCM(X, p = NULL, silent = FALSE, na.action = na.fail)
```

Arguments

X	numeric data.frame or matrix containing the data points as rows.
p	NULL or a number between 0 and 1 which specifies the fraction of hyperplanes to be used for subsampling. If p = 1, no subsampling is done. If p = NULL, the value of p is determined based on the size of the data set. See function ojaRank for details.
silent	logical, if subsampling is done or the expected computation time is too long, a warning message will be printed unless silent is TRUE. The default is FALSE.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.

Details

The function computes the Oja rank covariance matrix of the data set X, that is (since Oja ranks are centered) the covariance matrix of the Oja ranks of the data points in X, taken w.r.t. the data set X.

For a definition of the *Oja rank covariance matrix* and its properties see references below. The matrix X needs to have at least as many rows as columns in order to give sensible results. The return value is a quadratic, symmetric matrix having as many columns as X. It works also for matrices X with only one column and also vectors, but note that the variance of univariate ranks does not yield much information about the data.

The function offers a subsampling option in order to speed up computation for large data sets. The subsampling fraction is controlled by the parameter p. If p is not specified (which defaults to p = NULL), it is automatically determined based on the dimension of the problem. The function tries to realize a reasonable compromise between accuracy and computing time, that is, for sufficiently small data matrices X the sampling fraction p is set to 1. Subsampling is applied to hyperplanes, not data points. A sample is drawn once, all Oja ranks are then computed based on this sample. For further details on subsampling see function ojaRank. Subsampling is useful. Even for very small p useable results can be expected, see e.g. Example 2.

Value

a symmetric matrix with ncol(X) columns and rows.

Author(s)

Daniel Vogel

References

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92***(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).

Visuri, S., Koivunen, V., Oja, H. (1999), Sign and rank covariance matrices, J. Stat. Plann. Inference, ***91***, *557–575.*

Ollila, E., Croux, C., Oja, H. (2004), Influence function and asymptotic efficiency of the affine equivariant rank covariance matrix, Statistica Sinica, ***14***, *297–316.*

See Also

ojaRank, ojaSCM

Examples

```
### ----<< Example 1 >>---- : biochem data
data(biochem)
X <- biochem[,1:2]
ojaRCM(X)

# Oja ranks are centered
# (i.e. they add up to zero), and
# the following two return the same.
ojaRCM(X)
(1 - 1/nrow(X))*cov(ojaRank(X))



### ----<< Example 2 >>---- : 300 points in R^7
# The merit of subsampling.
# The following example might take a bit longer:

A <- matrix(c(1,0.5,1,4,2,0.5,-0.5,1,4), ncol = 3)
B <- A %x% A;  Sigma  <- (B %*% t(B))[1:7, 1:7]
# Sigma is some arbitrary positive definite matrix.
set.seed(123)
X <- rmvnorm(n = 300, sigma = Sigma)

cov2cor(Sigma) # the true correlation matrix
cor(X)  # Bravais-Pearson correlation
cov2cor(solve(ojaRCM(X)))
# correlation estimate based on Oja ranks
# The subsampling fraction in this example
# is p = 1.081438e-10.
# Yet it returns a sensible estimate.
```

ojaSCM *Oja Sign Convariance Matrix*

Description

The function computes the Oja sign covariance matrix of a data set X.

Usage

```
ojaSCM(X, center = "ojaMedian", p = NULL, silent = FALSE,
       na.action = na.fail, ...)
```

Arguments

X	numeric data.frame or matrix containing the data points as rows.
center	one of the following three: • a numeric vector giving the location of the data, • a function that computes a multivariate location (see details below) or • one of the following strings: – "colMean" (vector of means, function colMeans is called), – "ojaMedian" (function ojaMedian), – "spatialMedian" (function spatial.median from package ICSNP), – "compMedian" (marginal median) or – "HRMedian" (Hettmansperger and Randles median, function HR.Mest from package ICSNP). The default is "ojaMedian".
p	NULL or a number between 0 and 1 which specifies the fraction of hyperplanes to be used for subsampling. If p = 1, no subsampling is done. If p = NULL, the value of p is determined based on the size of the data set. See function ojaSign for details.
silent	logical, if subsampling is done or the expected computation time is too long, a warning message will be printed unless silent is TRUE. The default is FALSE.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.
...	arguments passed on to the location function.

Details

The function computes the Oja sign covariance matrix of the data set X, that is (if the Oja signs are centered by the Oja median) the covariance matrix of the Oja signs of the data points in X, taken w.r.t. X.

For a definition of the *Oja sign covariance matrix* and its properties see references below. The matrix `X` needs to have at least two columns and at least as many rows as columns in order to give sensible results. The return value is a quadratic, symmetric matrix having as many columns as `X`.

Oja signs (contrary to Oja ranks) require the computation of a centre (*location*) of the data cloud. The function offers various ways to specify the location. For details on location computation see function `ojaSign`.

The function offers a subsampling option in order to speed up computation for large data sets. The subsampling fraction is controlled by the parameter `p`. If `p` is not specified (which defaults to `p = NULL`), it is automatically determined based on the dimension of the problem. The function tries to realize a reasonable compromise between accuracy and computing time, that is, for sufficiently small data matrices `X` the sampling fraction `p` is set to 1. Subsampling is applied to hyperplanes, not data points. A sample is drawn once, all Oja signs are then computed based on this sample. For further details on subsampling see function `ojaSign`. Subsampling is useful. Even for very small `p` useable results can be expected, see e.g. Example 2.

Value

a symmetric matrix with `ncol(X)` columns and rows.

Author(s)

Daniel Vogel

References

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92****(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

Visuri, S., Koivunen, V., Oja, H. (1999), Sign and rank covariance matrices, J. Stat. Plann. Inference, ***91***, *557–575.*

Ollila, E., Oja, H., Croux, C. (2003), The affine equivariant sign covariance matrix: Asymptotic behavior and efficiencies, J. Multivariate Analysis, ***87***, *328–355.*

See Also

`ojaSign`, `ojaRCM`, `ojaMedian`, `spatial.median`, `HR.Mest`

Examples

```
### ----<< Example 1 >>---- : biochem data
data(biochem)
X <- biochem[,1:2]
ojaSCM(X)

# Oja signs are correctly centered
# (i.e. they add up to zero) when
# computed w.r.t. the Oja median
# Hence the following return the same,
ojaSCM(X, center = "ojaMedian", alg = "exact")
```

```
(1 - 1/nrow(X))*cov(ojaSign(X, alg = "exact"))
# but the following not.
ojaSCM(X, center = "colMean")
(1 - 1/nrow(X))*cov(ojaSign(X, center = "colMean"))



### ----<< Example 2 >>---- : 300 points in R^7
# The merit of subsampling.
# The following example might take a bit longer:

A <- matrix(c(1,0.5,1,4,2,0.5,-0.5,1,4), ncol = 3)
B <- A %x% A;  Sigma  <- (B %*% t(B))[1:7, 1:7]
# Sigma is some arbitrary positive definite matrix.
set.seed(123)
X <- rmvnorm(n=300,sigma=Sigma)

cov2cor(Sigma) # the true correlation matrix
cor(X)  # Bravais-Pearson correlation
cov2cor(solve(ojaSCM(X, center = "colMean")))
# correlation estimate based on Oja signs
# The subsampling fraction in this example
# is p = 4.542038e-09.
# Yet it returns a sensible estimate.
```

`ojaSign` *Oja Signs – Affine Equivariant Multivariate Signs*

Description

The function computes the Oja sign of a point `x` w.r.t. a data set `X` or, if no point `x` is given, the Oja signs of all points in `X`.

Usage

```
ojaSign(X, x = NULL, center = "ojaMedian", p = NULL, silent = FALSE,
        na.action = na.fail, ...)
```

Arguments

`X`	numeric data.frame or matrix containing the data points as rows.
`x`	`NULL` or a numeric vector, the point for which the Oja sign should be computed.
`center`	one of the following three: • a numeric vector giving the location of the data, • a function that computes a multivariate location (see details below) or • one of the following strings: – `"colMean"` (vector of means, function `colMeans` is called),

- "ojaMedian" (function ojaMedian),
- "spatialMedian" (function spatial.median from package ICSNP),
- "compMedian" (marginal median) or
- "HRMedian" (Hettmansperger and Randles median, function HR.Mest from package ICSNP).

The default is "ojaMedian".

p — NULL or a number between 0 and 1 which specifies the fraction of hyperplanes to be used for subsampling. If p = 1, no subsampling is done. If p = NULL, the value of p is determined based on the size of the data set. See details.

silent — logical, if subsampling is done or the expected computation time is too long, a warning message will be printed unless silent is TRUE. The default is FALSE.

na.action — a function which indicates what should happen when the data contain 'NA's. Default is to fail.

... — arguments passed on to the location function.

Details

The function computes the Oja sign of the point x w.r.t. to the data set X or, if no x is specified, the Oja signs of all data points in X w.r.t. X. For a definition of *Oja sign* see reference below.

The matrix X needs to have at least as many rows as columns in order to give sensible results. The vector x has to be of length ncol(X). If x is specified, a vector of length ncol(X) is returned. Otherwise the return value is a matrix of the same dimensions as X where the i-th row contains the Oja sign of the i-th row of X. The matrix X must have at least **two columns**. For univariate signs use sign.

Oja signs (contrary to Oja ranks) require the computation of a center (*location*) of the data cloud. The function offers various ways to do this. One can explicitly pass a location as a numeric vector (which has to be of length ncol(X)), or pass a function that computes a multivariate location.

This function must accept data sets of the same type as X (i.e. row-wise) and return a numeric vector of length ncol(X). For example center = colMeans will do, whereas center = mean will not, since mean(X) computes the (univariate) mean of all elements of the matrix X. Note that some location functions return a list also containing information on the data or the computation rather than the numeric location estimate only. Via ... arguments can be passed on to the location function, see example 2 below. The mean and several nonparametric location estimates are implemented and can also be specified by passing a corresponding string. The ... option is then available, too. Being based on the same nonparametric concept the Oja median is the natural location for Oja signs and is hence the default. But since it is the solution of an optimization problem, it may – depending on the optimization algorithm – return different values when run on the same data.

For large data sets the function offers a subsampling option in order to deliver (approximate) results within reasonable time. For n data points in R^k the computation of the Oja sign necessitates the evaluation of $N = choose(n, k-1)$ hyperplanes in R^k. If $p < 1$ is passed to the function, the computation is based on a random sample of only pN of all possible N hyperplanes. If p is not specified, it is automatically determined based on n and k to yield a sensible trade-off between accuracy and computing time. If $Nk^3 < 6 \cdot 10^6$, the sample fraction p is set to 1 (no subsampling). Otherwise p is chosen such that the computation (of one sign) *usually* takes around 20 seconds (on

a 1.66 GHz CPU and 1 GB RAM). If all Oja signs of X are requested, a hyperplane sample is drawn once, all Oja signs are then computed based on this sample.

Finally, subsampling is feasible. Even for very small p useable results can be expected, see e.g. the examples for the function ojaSCM.

Claudia Köllmann is acknowledged for bug-fixing this function.

Value

Either a numeric vector, the Oja sign of x, or a matrix of the same dimensions as X containing the Oja signs of X as rows.

Author(s)

Daniel Vogel, Jyrki Möttönen

References

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92***(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).

Oja, H. (1999), Affine invariant multivariate sign and rank tests and corresponding estimates: A review, Scand. J. Statist., ***26***, 319–343.

See Also

ojaRank, ojaSignedRank, ojaMedian, spatial.median, HR.Mest, ojaSCM

Examples

```
### ----<< Example 1 >>---- : 30 points in R^2
set.seed(123)
X <- rmvnorm(n = 30, mean = c(0,0)) # from package 'mvtnorm'
y <- c(100,100)
om <- ojaMedian(X, alg = "exact")

ojaSign(X)
ojaSign(X,y)
# possible ways of specifying the mean as location:
ojaSign(X, center = "colMean")
ojaSign(X, center = colMeans)
ojaSign(X, center = colMeans(X))

# The following two return the same (only in different time),
ojaSign(X, center = colMeans)
t(apply(X, 1, function(y){ojaSign(X, y, center = colMeans)}))

# but the following not (due to different subsampling).
# 1)
set.seed(123)
ojaSign(X, center = colMeans, p = 0.9, silent = TRUE)
```

```
# 2)
set.seed(123)
t(apply(X, 1, function(y){ojaSign(X, y, c = colMeans,p = 0.9, s = TRUE)}))
# In 1) one subsample for all signs is drawn, whereas in 2)
# a different sample for each sign is drawn.

### ----<< Example 2 >>---- : Oja median
# The Oja sign of the Oja median is zero:
ojaSign(X, x = om, alg = "exact")
# The default location function 'ojaMedian()'
# is called with method "exact",
# which gives the same result as:
ojaSign(X, x = om, center = om)
# But note: The following is likely to not return zero.
ojaSign(X, x = ojaMedian(X))
# The default method of 'ojaMedian()' is "evo",
# which is fast, but returns approximate results.



### ----<< Example 3 >>---- : 400 points in R^6
# Subsampling is done.
# The following example might take a bit longer:

set.seed(123)
X <- rmvnorm(n = 400, mean = rep(0, 6))
os1 <- ojaSign(X, x = 1:6, c = colMeans)
# Note: the following command may take several minutes
os2 <- ojaSign(X, x = 1:6, p = 0.0000001, c = colMeans)
```

`ojaSignedRank` *Oja Signed Ranks – Affine Equivariant Multivariate Signed Ranks*

Description

The function computes the Oja signed rank of a point `x` w.r.t. a data set `X` or, if no point `x` is given, the Oja signed ranks of all points in `X`.

Usage

```
ojaSignedRank(X, x = NULL, p = NULL, silent = FALSE,
              na.action = na.fail)
```

Arguments

`X`	numeric data.frame or matrix containing the data points as rows.
`x`	`NULL` or a numeric vector, the point for which the Oja signed rank should be computed.

p	NULL or a number between 0 and 1 which specifies the fraction of hyperplanes to be used for subsampling. If p = 1, no subsampling is done. If p = NULL, the value of p is determined based on the size of the data set. See details.
silent	logical, if subsampling is done or the expected computation time is too long, a warning message will be printed unless silent is TRUE. The default is FALSE.
na.action	a function which indicates what should happen when the data contain 'NA's. Default is to fail.

Details

The function computes the Oja signed rank of the point x w.r.t. the data set X or, if no x is specified, the Oja signed ranks of all data points in X w.r.t. X. For a definition of *Oja signed rank* see Hettmansperger et al. (1997) formula (9).

The matrix X needs to have at least as many rows as columns in order to give sensible results. The vector x has to be of length ncol(X). If x is specified, a vector of length ncol(X) is returned. Otherwise the return value is a matrix of the same dimensions as X where the i-th row contains the Oja rank of the i-th row of X.

The function will also work for matrices X with only one column and also vectors. Then (univariate) signed ranks are returned.

For $n = nrow(X)$ data points in R^k, where $k = ncol(X)$, the computation of the Oja signed rank necessitates the evaluation of $N = 2^k * choose(n, k)$ hyperplanes in R^k. Thus for large data sets the function offers a subsampling option in order to deliver (approximate) results within reasonable time. The subsampling fraction is controlled by the parameter p: If $p < 1$ is passed to the function, the computation is based on a random sample of only pN of all possible N hyperplanes. If p is not specified (which defaults to p = NULL), it is automatically determined based on n and k to yield a sensible trade-off between accuracy and computing time. If $Nk^3 < 6 \cdot 10^6$, the sample fraction p is set to 1 (no subsampling). If all Oja signed ranks of X are requested, a hyperplane sample is drawn once and all Oja signed ranks are then computed based on this sample.

Finally, subsampling is feasible. Even for very small p useable results can be expected, see e.g. the examples for the function ojaRCM.

Value

Either a numeric vector, the Oja signed rank of x, or a matrix of the same dimensions as X containing the Oja signed ranks of X as rows.

Author(s)

Jyrki Möttönen

References

Fischer D, Mosler K, Möttönen J, Nordhausen K, Pokotylo O and Vogel D (2020). "Computing the Oja Median in R: The Package OjaNP." Journal of Statistical Software, ***92****(8), pp. 1-36. doi: 10.18637/jss.v092.i08 (URL: http://doi.org/10.18637/jss.v092.i08).*

Hettmansperger, T. P.,\ Möttönen, J. and Oja, H. (1997), Affine invariant multivariate one-sample signed-rank tests, J.\ Amer. Statist. Assoc., ***92****, 1591–1600.*

Oja, H. (1999), Affine invariant multivariate sign and rank tests and corresponding estimates: A review, Scand. J. Statist., ***26***, *319–343.*

See Also

ojaSign, ojaRank, ojaRCM, hyperplane

Examples

```
set.seed(123)
X <- rmvnorm(n = 30,mean = c(0,0)) # from package 'mvtnorm'
ojaSignedRank(X)
ojaSignedRank(X, x = c(0,0)) # zero
```

Index

∗ **datasets**
biochem, 3
∗ **htest**
oja1sampleTest, 5
ojaCsampleTest, 7
∗ **misc**
ojaMedianControl, 11
∗ **multivariate**
hyperplane, 4
oja1sampleTest, 5
ojaCsampleTest, 7
ojaMedian, 9
ojaMedianFn, 13
ojaRank, 14
ojaRCM, 16
ojaSCM, 19
ojaSign, 21
ojaSignedRank, 24
∗ **nonparametric**
hyperplane, 4
oja1sampleTest, 5
ojaCsampleTest, 7
ojaMedian, 9
ojaMedianFn, 13
ojaRank, 14
ojaRCM, 16
ojaSCM, 19
ojaSign, 21
ojaSignedRank, 24
∗ **package**
OjaNP-package, 2

biochem, 3

colMeans, *19*, *21*
cov, *12*
cov4, *12*

HR.Mest, *19, 20*, *22, 23*
hyperplane, 4, *15*, *26*

ics, *12*

oja1sampleTest, 5, *9*
ojaCsampleTest, 7
ojaMedian, *6*, *8*, 9, *11*, *13, 14*, *19, 20*, *22, 23*
ojaMedianControl, *9*, 11
ojaMedianEvo *(*ojaMedian*)*, 9
ojaMedianEx *(*ojaMedian*)*, 9
ojaMedianExB *(*ojaMedian*)*, 9
ojaMedianFn, 13
ojaMedianGrid *(*ojaMedian*)*, 9
OjaNP-package, 2
ojaRank, *7–9*, 14, *17, 18*, *23*, *26*
ojaRCM, *15*, 16, *20*, *25, 26*
ojaSCM, *18*, 19, *23*
ojaSign, *5–9*, *15*, *19, 20*, 21, *26*
ojaSignedRank, *5, 6*, *15*, *23*, 24

sign, *22*
spatial.median, *19, 20*, *22, 23*