Package 'mrfDepth'

May 9, 2026

Type Package

Version 1.0.17

Date 2024-05-23

Title Depth Measures in Multivariate, Regression and Functional Settings

Description Tools to compute depth measures and implementations of related tasks such as outlier detection, data exploration and classification of multivariate, regression and functional data.

Depends R (>= 3.6.0), ggplot2

Imports abind, geometry, grid, matrixStats, reshape2, Rcpp (>= 0.12.6)

Suggests robustbase

LinkingTo RcppEigen (>= 0.3.2.9.0), Rcpp (>= 0.12.6), RcppArmadillo (>= 0.7.600.1.0)

License GPL (>= 2)

LazyLoad yes

RoxygenNote 7.1.2

NeedsCompilation yes

Author Pieter Segaert [aut],
Mia Hubert [aut],
Peter Rousseeuw [aut],
Jakob Raymaekers [aut, cre],
Kaveh Vakili [ctb]

Maintainer Jakob Raymaekers <jakob.raymaekers@kuleuven.be>

Repository CRAN

Date/Publication 2024-05-24 21:20:02 UTC

Contents

adjOutl *Adjusted outlyingness of points relative to a dataset*

Description

Computes the skew-adjusted outlyingness of p-dimensional points z relative to a p-dimensional dataset x. For each multivariate point z_i, its adjusted outlyingness relative to x is defined as its maximal univariate adjusted outlyingness measured over all directions. To obtain the univariate adjusted outlyingness in the direction v, the dataset x is projected on v, and the robustly skew-adjusted standardized distance of $v'z_i$ to the median of the projected data points $\mathtt{x}v$ is computed.

Usage

```
adjOutl(x, z = NULL, options = list())
```

Arguments

x	An n by p data matrix.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the adjusted outlyingness. If z is not specified, it is set equal to x.
options	A list of available options:

- type
  Determines the desired type of invariance and should be one of "Affine", "Rotation" or "Shift". When the option "Affine" is used, the directions v are orthogonal to hyperplanes spanned by p observations from x. When the option "Rotation" is used, the directions pass by two randomly selected observations from x. With the option "Shift", directions are randomly generated.
  Defaults to "Affine".
- ndir
  Determines the number of directions v by setting ndir to a specific number or to "all". In the latter case, an exhaustive search over all possible directions (according to type) is performed. If ndir is larger than the number of possible directions, the algorithm will automatically use this setting.
  Defaults to $250p$ when type="Affine", to 5000 when type="Rotation" and to 12500 when type="Shift".
- seed
  A strictly positive integer specifying the seed to be used by the C++ code.
  Defaults to 10.

Details

The adjusted outlyingness (AO) of multivariate data was introduced in Brys et al. (2005) and studied in more detail in Hubert and Van der Veeken (2008). It extends the Stahel-Donoho outlyingness towards skewed distributions.

Depending on the dimension p, different approximate algorithms are implemented. The affine invariant algorithm can only be used when $n > p$. It draws `ndir` times at random p observations from `x` and considers the direction orthogonal to the hyperplane spanned by these p observations. At most p out of n directions can be considered. The orthogonal invariant version can be applied to high-dimensional data. It draws `ndir` times at random 2 observations from `x` and considers the direction through these two observations. Here, at most 2 out of n directions can be considered. Finally, the shift invariant version randomly draws `ndir` vectors from the unit sphere.

The resulting AO values are invariant to affine transformations, rotations and shifts respectively provided that the `seed` is kept fixed at different runs of the algorithm. Note that the AO values are guaranteed to increase when more directions are considered provided the seed is kept fixed, as this ensures that the random directions are generated in a fixed order.

An observation from `x` and `z` is flagged as an outlier if its AO exceeds a cutoff value. This cutoff value is determined using the procedure in Rousseeuw et al. (2018). First, the logarithm of the AO values is taken to render their distribution more symmetric, after which a normal approximation yields a cutoff on these values. The cutoff is then transformed back by applying the exponential function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it. Furthermore, the univariate adjusted outlyingness of the projected points $\mathtt{x}v$ is ill-defined when the scale in its denominator becomes zero. This can happen when many observations collapse. In these cases the algorithm will stop and give a warning. The returned values then include the direction v as well as an indicator specifying which of the observations of `x` belong to the hyperplane orthogonal to v.

This function extends the `adjOutlyingness` function in the package `robustbase`. It has more options for choosing the directions, it allows to compute the adjusted outlyingness of points not belonging to the data matrix `x` and it is faster as it is fully implemented in C++. On the other hand, the constants (3 and -4) used in the definition of the adjusted outlyingness can not be modified in this implementation.

Value

A list with components:

`outlyingnessX` Vector of length n giving the adjusted outlyingness of the observations in `x`.

`outlyingnessZ` Vector of length m giving the adjusted outlyingness of the points in `z` relative to `x`.

`cutoff` Points whose adjusted outlyingness exceeds this cutoff can be considered as outliers with respect to `x`.

`flagX` Observations of `x` whose adjusted outlyingness exceeds the cutoff receive a flag `FALSE`, regular observations receive a flag `TRUE`.

`flagZ` Points of `z` whose adjusted outlyingness exceeds the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.

`singularSubsets`
When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p - 1$. In such a case the orthogonal direction can not be uniquely determined. This is an indication that

the data are not in general position. When the input parameter type is equal to "Rotation" it is possible that two randomly selected points of the data coincide due to ties in the data. In such a case this value signals how many times this happens.

`dimension` When the data x are lying in a lower dimensional subspace, the dimension of this subspace.

`hyperplane` When the data x are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust skew-adjusted scale of xv is equal to zero, this equals v.

`inSubspace` When a direction v is found such that AO(xv) is ill-defined, the observations from x which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

P. Segaert using C++ code by K. Vakili, P. Segaert, G. Brys and M. Maechler.

References

Brys G., Hubert M., Rousseeuw P.J. (2005). A robustification of Independent Component Analysis. *Journal of Chemometrics*, **19**, 364–375.

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Hubert M., Vandervieren E. (2008). An adjusted boxplot for skewed distributions. *Computational Statistics & Data Analysis*, **52**, 5186–5201.

Rousseeuw P.J., Raymaekers J., Hubert M., (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

sprojdepth, sprojmedian, dirOutl, outlyingness

adjbox, adjOutlyingness from package robustbase.

Examples

```
# Compute the adjusted outlyingness of a simple
# two-dimensional dataset. Outliers are plotted
# in red.
data(geological)
BivData <- geological[c("MnO","MgO")]
Result <- adjOutl(x = BivData)
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")

# The number of directions may be specified through
# the option list. The resulting adjusted outlyingness
```

```
# is monotone increasing in the number of directions.
Result1 <- adjOutl(x = BivData,
                   options = list(ndir = 50)
                   )
Result2 <- adjOutl(x = BivData,
                   options = list(ndir = 100)
                   )
which(Result2$outlyingnessX - Result1$outlyingnessX < 0)
# This is however not the case when the seed is changed
Result1 <- adjOutl(x = BivData,
                   options = list(ndir = 50)
                   )
Result2 <- adjOutl(x = BivData,
                   options = list(ndir = 100,
                                  seed = 950)
                   )
plot(Result2$outlyingnessX - Result1$outlyingnessX,
     xlab = "Index", ylab = "Difference in AO")

# We can also consider directions through two data
# points. If the sample is small enough one may opt
# to search over all choose(n,2) directions.
# Note that the computational load increases dramatically
# as n becomes larger.
data(bloodfat)
BivData <- bloodfat[1:100,] # Consider a small toy example.
Result <- adjOutl(x = BivData,
                  options = list(type = "Rotation",
                                 ndir = "all")
                  )
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")

# Alternatively one may consider randomly generated directions.
data(bloodfat)
Result <- adjOutl(x = bloodfat,
                  options = list(type = "Shift",
                                 ndir = 1000)
                  )
IndOutliers <- which(!Result$flagX)
plot(bloodfat)
points(bloodfat[IndOutliers,], col = "red")
```

`bagdistance` *Bagdistance of points relative to a dataset*

Description

Computes the bagdistance of p-dimensional points `z` relative to a p-dimensional dataset `x`. To compute the bagdistance of a point z_i first the bag of `x` is computed as the depth region containing

the 50% observations (of x) with largest halfspace depth. Next, the ray from the halfspace median θ through z_i is considered and c_z is defined as the intersection of this ray and the boundary of the bag. The bagdistance of z_i to x is then given by the ratio between the Euclidean distance of z_i to the halfspace median and the Euclidean distance of c_z to the halfspace median.

Usage

```
bagdistance(x, z = NULL, options = list())
```

Arguments

x	An n by p data matrix.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the bagdistance. If z is not specified, it is set equal to x.
options	A list of available options: • approx In two dimensions one may choose to use an approximate algorithm or the exact algorithm to find the bag. Defaults to TRUE. • max.iter The maximum number of steps in the bisection algorithm to find the intersection point c_z (see Details). Defaults to 100. • All options may be specified that are passed to the hdepth function, see hdepth for details. Note that the option parameter approx is by default set to TRUE to save computation time.

Details

The bagdistance has been introduced in Hubert et al. (2015) and studied in Hubert et al. (2017). It does not assume symmetry and is affine invariant. Note that when the halfspace is not computed in an affine invariant way, the bagdistance cannot be affine invariant either.

The function first computes the halfspace depth and the halfspace median of x. Additional options may be passed to the hdepth routine by specifying them in the option list argument.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Depending on the dimensions different algorithms are used. For $p = 1$ the bagdistance is computed exactly. For $p = 2$ the default setting (options$approx=TRUE) uses an approximated algorithm. Exact computation, based on the exact algoritm to compute the contours of the bag (see the depthContour function), is obtained by setting options$approx to FALSE. Note that this may lead to an increase in computation time.

For the approximated algorithm, the intersection point c_z is approximated by searching on each ray the point whose depth is equal to the median of the depth values of x. As the halfspace depth is monotone decreasing along the ray, a bisection algorithm is used. Starting limits are obtained by projecting the data on the direction and considering the data point with univariate depth corresponding to the median of the halfspace depths of x. By definition the multivariate depth of this point has to be lower or equal than its univariate depth. A second limit is obtained by considering the deepest

location estimate. The maximum number of iterations bisecting the current search interval can be specified through the options argument `max.iter`.

An observation from `z` is flagged as an outlier if its bagdistance exceeds a cutoff value. This cutoff is equal to the squareroot of the 0.99 quantile of the chi-squared distribution with p degrees of freedom.

Value

A list with components:

`bagdistance`	The bagdistance of the points of `z` with respect to the data matrix `x`.
`cutoff`	Points of `z` whose bagdistance exceeds this cutoff can be considered as outliers with respect to `x`.
`flag`	Points of `z` whose bagdistance exceeds the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.
`converged`	Vector of length `m` indicating for each point of `z` whether the bisection algorithm converged within the maximum number of steps specified by `max.iter` in the `options` list.
`dimension`	When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.
`hyperplane`	When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert.

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection. *Statistical Methods & Applications*, **24**, 177–202.

Hubert M., Rousseeuw P.J., Segaert P. (2017). Multivariate and functional classification using depth and distance. *Advances in Data Analysis and Classification*, **11**, 445–466.

See Also

`depthContour`, `hdepth`, `bagplot`

Examples

```
# Generate some bivariate data
set.seed(5)
nObs <- 500
XS <- matrix(rnorm(nObs * 2), nrow = nObs, ncol = 2)
A <- matrix(c(1,1,.5,.1), ncol = 2, nrow = 2)
X <- XS %*% A
```

```
# In two dimensions we may either use the approximate
# or the exact algorithm to compute the bag.
respons.exact <- bagdistance(x = X, options = list(approx = FALSE))
respons.approx <- bagdistance(x = X, options = list(approx = TRUE))
# Both algorithms yield fairly similar results.
plot(respons.exact$bagdistance, respons.approx$bagdistance)
abline(a = 0, b = 1)

# In Hubert et al. (2015) it was shown that for elliptical
# distributions the squared bagdistance relates to the
# squared Mahalanobis distances. This may be easily illustrated.
mahDist <- mahalanobis(x = X, colMeans(X), cov(X))
plot(respons.exact$bagdistance^2, mahDist)

# Computation of the bagdistance relies on the computation
# of halfspace depth using the hdepth function. Options for
# the hdepth routine can be passed down using the options
# arguments. Note that the bagdistance is only affine invariant
# if the halfspace depth is computed in an affine invariant way.
options <-list(type = "Rotation",
               ndir = 375,
               approx = TRUE,
               seed = 78341)
respons.approx.rot <- bagdistance(x = X, options = options)
plot(respons.exact$bagdistance, respons.approx.rot$bagdistance)
abline(a = 0, b = 1)
```

`bagplot` *Draws a bagplot, a bivariate boxplot*

Description

This function draws a bagplot of bivariate data, based on the result of a call to `compBagplot`. The bagplot is a generalisation of the univariate boxplot to bivariate data. It aims to visualize the location, spread, skewness and outliers of the data set.

Usage

```
bagplot(compbag.result,
        colorbag = NULL, colorloop = NULL, colorchull = NULL,
        databag = TRUE, dataloop = TRUE, plot.fence = FALSE)
```

Arguments

`compbag.result`	The return of a call to `compBagplot`.
`colorbag`	The color of the bag (which contains the 50% observations with largest depth).
`colorloop`	The color of the loop (which contains the regular observations).
`colorchull`	When the bagplot is based on halfspace depth, the depth region with maximal depth is plotted. This argument controls its color.

databag	Logical indicating whether data points inside the bag need to be plotted. Defaults to TRUE.
dataloop	Logical indicating whether data points inside the fence need to be plotted. Defaults to TRUE.
plot.fence	Logical indicating whether the fence should be plotted. Defaults to FALSE.

Details

The bagplot has been proposed by Rousseeuw et al. (1999) as a generalisation of the boxplot to bivariate data. It is constructed based on halfspace depth and as such is invariant under affine transformations. Similar graphical representations can be obtained by means of other depth functions, as illustrated in Hubert and Van der Veeken (2008) and in Hubert et al. (2015). See compBagplot for more details.

The deepest point is indicated by a red diamond symbol, the outlying observations by red stars.

The plot is made using ggplot2. The plot itself is returned by the function and is fully customisable using standard ggplot2 commands.

Author(s)

P. Segaert

References

Rousseeuw P.J., Ruts I., Tukey J.W. (1999). The bagplot: a bivariate boxplot. *The American Statistician*, **53**, 382–387.

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Hubert M., Rousseeuw P.J., Segaert P. (2015). Rejoinder of 'Multivariate functional outlier detection'. *Statistical Methods & Applications*, **24**, 269–277.

See Also

compBagplot, hdepth, projdepth, sprojdepth, dprojdepth.

Examples

```
data(bloodfat)

# The bagplot can be plotted based on halfspace depth, projection depth,
# skewness-adjusted projection depth or directional projection depth.
# Note that projection depth is not appropiate for skewed data.
# bagplot(compBagplot(bloodfat))
bagplot(compBagplot(bloodfat, type = "projdepth"))
bagplot(compBagplot(bloodfat, type = "dprojdepth"))

# The main features of the bagplot can easily be adjusted.
result <- compBagplot(bloodfat, type = "projdepth")
bagplot(result, databag = FALSE, dataloop = FALSE)
```

```
bagplot(result, colorbag = rgb(0.2, 0.2, 0.2), colorloop = "lightgreen")

data(cardata90)
result <- compBagplot(cardata90, type = "projdepth")
bagplot(result)

# Compared to the original paper on the bagplot,
# an additional outlier is identified. However this
# point lies very close to the fence and this may be
# attributed to differences in numerical rounding.
# This may be illustrated by plotting the fence.
plot <- bagplot(result, plot.fence = TRUE)
plot

# The returned object is a ggplot2 object and may be
# edited using standard ggplot2 commands.
library("ggplot2")
plot + ylab("Engine displacement") + xlab("Weight in pounds")
```

`bloodfat` *Blood data for patients with narrowing arteries*

Description

Data were collected on the concentration of plasma cholesterol and plasma triglycerides (mg/dl) for 371 male patients evaluated for chest pain. For 51 of those patients, no evidence of heart disease was found. This subset corresponds to the remaining 320 patients for which there was evidence of narrowing arteries.

Usage

```
data(bloodfat)
```

Format

A data frame containing the following variables:

`Cholesterol` Concentration of plasma cholesterol [mg/dl].

`Triglycerides` Concentration of plasma triglycerides [mg/dl].

Source

Hand D.J., Daly F., Lunn A., McConway A. (1994). A Handbook of Small Data Sets. *Londen: Chapman and Hall*, dataset 277.

References

Scott D.W., Gotto A.M., Cole J.S., Gorry G.A. (1978). Plasma lipids as collateral risk factors in coronary artery disease: a study of 371 males with chest pain. *Journal of Chronic Diseases*, **31**, 337–345.

Examples

```
data(bloodfat)
plot(bloodfat)
```

`cardata90` *Car data from Consumer Reports in 1990*

Description

Subset from data on cars taken from pages 235-255, 281-285 and 287-288 of the April 1990 Consumer Reports Magazine.

Usage

```
data(cardata90)
```

Format

A data frame containing the following variables:

`Weight` Weight of the car (in pounds).

`Disp` Engine displacement (in cubic inches).

Source

Consumer Reports, April 1990, 235–288.

Chambers J.M., Hastie T.J. (1993). Statistical Models in S. *Londen: Chapman and Hall*, 46–47.

References

Rousseeuw P.J., Ruts I., Tukey J.W. (1999). The bagplot: a bivariate boxplot. *The American Statistician*, **53**, 382–387.

Examples

```
data(cardata90)
plot(cardata90)
```

characterA *Writing trajectories of the letter 'a'*

Description

Subset of the 'Character Trajectories Data Set' from the UCI Machine Learning Repository. The data set consists of trajectories of the tip of a pen whilst writing the letter 'a'. All samples are from the same writer. Original data has been processed.

Usage

```
data(characterA)
```

Format

Three dimensional array. The first dimension represents time. The second dimension corresponds to the observation number. The third dimension contains the X and Y coordinates.

Source

Bache K., Lichman M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

References

Williams B.H., Toussaint M., Storkey A.J. (2006). Extracting motion primitives from natural handwriting data. In ICANN, volume 2, pages 634–643.

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Examples

```
data(characterA)
par(mfrow = c(1,2))
matplot(y = characterA[,,1],
        type = "l", col = "black", lty = 1,
        xlab = "Time", ylab = "X position of the pen")
matplot(y = characterA[,,2],
        type = "l", col = "black", lty = 1,
        xlab = "Time", ylab = "Y position of the pen")
par(mfrow = c(1,1))
```

characterI *Writing trajectories of the letter i*

Description

Subset of the 'Character Trajectories Data Set' from the UCI Machine Learning Repository. The data set consists of trajectories of the tip of a pen whilst writing the letter 'i'. All samples are from the same writer. Original data has been processed.

Usage

```
data(characterI)
```

Format

Three dimensional array. The first dimension represents time. The second dimension corresponds to the observation number. The third dimension contains the X and Y coordinates.

Source

Bache K., Lichman M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

References

Williams B.H., Toussaint M., Storkey A.J. (2006). Extracting motion primitives from natural handwriting data. In ICANN, volume 2, pages 634–643.

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Examples

```
data(characterI)
par(mfrow = c(1,2))
matplot(y = characterI[,,1],
        type = "l", col = "black", lty = 1,
        xlab = "Time", ylab = "X position of the pen")
matplot(y = characterI[,,2],
        type = "l", col = "black", lty = 1,
        xlab = "Time", ylab = "Y position of the pen")
par(mfrow = c(1,1))
```

`cmltest` *Test for linearity of the conditional median in simple regression*

Description

A test based on regression depth for linearity of the conditional median `z` given the simple regression dataset `x`. The computation of the regression depth of `z` with respect to `x` is done by the function `rdepth`. The test is only valid when `x` contains no duplicates.

Usage

```
cmltest(x, z)
```

Arguments

`x`	An n by 2 regression data matrix. The first column is the explanatory variable, the second column corresponds to the response variable.
`z`	A matrix with one row containing an intercept and slope.

Details

The following hypothesis test is performed:
H_0: The data come from a model with: $med(x_2|x_1) = z_1 + z_2 * x_1$
The test statistic being used is the regression depth of `z` with respect to `x`.

Value

`pval`	The p-value of the hypothesis test.

Author(s)

P. Segaert

References

Van Aelst S., Rousseeuw P.J., Hubert M., Struyf A. (2002). The deepest regression method. *Journal of Multivariate Analysis*, **81**, 138–166.

Rousseeuw P.J., Struyf A. (2002). A depth test for symmetry. In: *Goodness-of-Fit Tests and Model Validity*, Birkhäuser Boston, pages 401–412.

See Also

`rdepth`, `rdepthmedian`

Examples

```
data(stars)

# Compute the least squares fit. Due to outliers
# this fit will be bad and thus H0 should be rejected.
temp <- lsfit(x = stars[,1], y = stars[,2])$coefficients
intercept <- temp[1]
slope <- temp[2]
z <- matrix(c(intercept, slope), nrow = 1)
pvalue1 <- cmltest(x = stars[!duplicated(stars), ], z = z)
pvalue1

# Let's now test the deepest regression line.
result <- rdepthmedian(x = stars)
pvalue2 <- cmltest(x = stars[!duplicated(stars), ], z = matrix(result$deepest, nrow = 1))
pvalue2

plot(stars)
abline(a = intercept, b = slope)
abline(result$deepest, col = "red")
text(x = 3.8, y = 5.3, labels = paste("p-value", round(pvalue1, digits = 3)))
text(x = 4.45, y = 4.8, labels = paste("p-value", round(pvalue2, digits = 3)),
     col = "red")
```

`compBagplot` *Computations for drawing a bagplot*

Description

Computes all elements of the bagplot, a generalisation of the univariate boxplot to bivariate data. The bagplot can be computed based on halfspace depth, projection depth, skewness-adjusted projection depth and directional projection depth. To draw the actual plot, the function `bagplot` needs to be called on the result of `compBagplot`.

Usage

```
compBagplot(x, type = "hdepth", sizesubset = 500,
            extra.directions = FALSE, options = NULL)
```

Arguments

`x`	An n by 2 data matrix.
`type`	Determines the depth function used to construct the bagplot: `"hdepth"` for halfspace depth, `"projdepth"` for projection depth, `"sprojdepth"` for skewness-adjusted projection depth and `"dprojdepth"` for directional projection depth. Defaults to `"hdepth"`.
`sizesubset`	When computing the bagplot based on halfspace depth, the size of the subset used to perform the main computations. See Details for more information. Defaults to 500.

`extra.directions`
: Logical indicating whether additional directions should be considered in the computation of the fence for the bagplot based on projection depth or skewness-adjusted projection depth. If set to `TRUE` an additional 250 equispaced directions are added to the directions defined by the points in `x` themselves and the center. If `FALSE` only directions determined by the points in `x` are considered.
Defaults to `FALSE`.

`options`
: A list of options to pass to the `projdepth`, `sprojdepth` or `dprojdepth` function. In addition the following option may be specified:

  - `max.iter`
  The maximum number of iterations in the bisection algorithm used to compute the depth contour corresponding to the cutoff. See `depthContour` for more information.
  Defaults to 100.

Details

The bagplot has been proposed by Rousseeuw et al. (1999) as a generalisation of the boxplot to bivariate data. It is constructed based on halfspace depth. In the original format the deepest point is indicated by a "+" and is contained in the bag which is defined as the depth region containing the 50% observations with largest depth. The fence is obtained by inflating the bag (relative to the deepest point) by a factor of three. The loop is the convex hull of the observations of `x` inside the fence. Observations outside the fence are flagged as outliers and plotted with a red star. This function only computes all the components constituting the bagplot. The bagplot itself can be drawn using the `bagplot` function.

The bagplot may also be defined using other depth functions. When using projection depth, skewness-adjusted projection depth or directional projection depth, the bagplot is build as follows. The center corresponds to the observation with largest depth. The bag is constructed as the convex hull of the fifty percent points with largest depth. Outliers are identified as points with a depth smaller than a cutoff value, see `projdepth`, `sprojdepth` and `dprojdepth`for the precise definition. The loop is computed as the convex hull of the non-outlying points. The fence is approximated by the convex hull of those points that lie on rays from the center through the vertices of the bag and have a depth that equals the cutoff depth. For a better approximation the user can set the input parameter `extraDirections` to `TRUE` such that an additional 250 equally spaced directions on the circle are considered.

The computation of the bagplot based on halfspace depth can be time consuming. Therefore it is possible to limit the bulk of the computations to a random subset of the data. Computations of the halfspace median and the bag are then based on this random subset. The number of points in this subset can be controlled by the optional argument `sizesubset`.

It is first checked whether the data is found to lie on a line. If so, the routine will give a warning, giving back the dimension of the subspace (being 1) together with the normal vector to that line.

Value

A list with components:

center	Center of the data. When type = "hdepth", this corresponds with the Tukey median. In other cases this point corresponds to the point with maximal depth.
chull	When type = "hdepth", these are the vertices of the region with maximal halfspace depth. In other cases this is a null vector.
bag	The coordinates of the vertices of the bag.
fence	The coordinates of the vertices of the fence.
datatype	An n by 3 matrix. The first two columns correspond with x. The third column indicates the position of each observation of x in the bagplot: 2 for observations in the bag, 1 for the observations in the fence and 3 for outliers. Note that points may not be in the same order as in x.
flag	A vector of length n wich is 0 for outliers and 1 for regular observations of x.
depth	The depth of the observations of x.
dimension	If the data are lying in a lower dimensional subspace, the dimension of this subspace.
hyperplane	If the data are lying in a lower dimensional subspace, a direction orthogonal to this subspace.
type	Same as the input parameter type.

Author(s)

P. Segaert based on Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf.

References

Rousseeuw P.J., Ruts I., Tukey J.W. (1999). The bagplot: A bivariate boxplot. *The American Statistician*, **53**, 382–387.

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Hubert M., Rousseeuw P.J., Segaert, P. (2015). Rejoinder to 'Multivariate functional outlier detection'. *Statistical Methods & Applications*, **24**, 269–277.

See Also

bagplot, hdepth, projdepth, sprojdepth, dprojdepth.

Examples

```
data(bloodfat)
# Result <- compBagplot(bloodfat)
# bagplot(Result)

# The sizesubset argument may be used to control the
# computation time when computing the bagplot based on
# halfspace depth. However results may be unreliable when
# choosing a small subset for the main computations.
# system.time(Result1 <- compBagplot(bloodfat))
```

```
# system.time(Result2 <- compBagplot(bloodfat, sizesubset = 100))
# bagplot(Result1)
# bagplot(Result2)

# When using any of the projection depth functions,
# a list of options may be passed down to the corresponding
# outlyingness routines.
options <- list(type = "Rotation",
                ndir = 50,
                stand = "unimcd",
                h = floor(nrow(bloodfat)*3/4))
Result <- compBagplot(bloodfat,
                      type = "projdepth", options = options)
bagplot(Result)

# The fence is computed using the depthContour function.
# To get a smoother fence, one may opt to consider extra
# directions.
options <- list(ndir = 500,
                seed = 36)
Result <- compBagplot(bloodfat,
                      type = "dprojdepth", options = options)
bagplot(Result, plot.fence = TRUE)

options <- list(ndir = 500,
                seed = 36)
Result <- compBagplot(bloodfat,
                      type = "dprojdepth", options = options,
                      extra.directions = TRUE)
bagplot(Result, plot.fence = TRUE)
```

---

depthContour *Depth contours of multivariate data*

Description

Computes the vertices of depth contours of multivariate data. The contours can be computed based on halfspace depth, projection depth, skewness-adjusted projection depth or directional projection depth. To make the actual plot for bivariate data, the function `plotContours` needs to be called on the result of `depthContour`.

Usage

```
depthContour(x, alpha = NULL, type = "hdepth", directions = NULL, options = NULL)
```

Arguments

x	An n by p data matrix.
alpha	A vector containing the depth values of which the depth contours have to be computed.

`type` The depth used in the computation of the contours: `hdepth` for halfspace depth, `projdepth` for projection depth, `sprojdepth` for skewness-adjusted projection depth and `dprojdepth` for directional projection depth.
Defaults to `hdepth`.

`directions` An m by p matrix specifying the directions on which to compute the vertices (see Details).

`options` A list of options to pass to `hdepth`, `projdepth`, `sprojdepth` or `dprojdepth`. In addition the following option may be specified:

- `max.iter`
  The maximum number of iterations in the bisection algorithm used to compute the depth contour corresponding to level alpha.
  Defaults to 100.

Details

Depth contours of level α (or α-depth contours) are the boundaries of depth regions. Depth regions of level α are defined as regions in space containing the multivariate points whose depth value is at least α.

For bivariate data halfspace depth contours can be computed exactly following the algorithm in Ruts and Rousseeuw (1996). When the data are not in general position (i.e. when there is a line containing more than two observations) dithering is performed by adding random Gaussian noise to the data.

In all other cases an approximated method is performed using a bisection algorithm. Intersections with the depth contours are searched on lines originating from the depth median. The user can specify a set of directions corresponding to these lines. By default a random set of $250p$ directions is considered. On each direction a point is searched having depth α. Starting limits are obtained by projecting the data on the direction and considering the data point with univariate depth α. By definition the multivariate depth of this point has to be lower or equal to α. A second limit is obtained by considering the deepest location estimate. The maximum number of iterations bisecting the current search interval can be specified through the options argument `max.iter`. Note that this method is only affine or rotation equivariant if the chosen directions are affine or rotation equivariant.

It is first checked whether the data is found to lie in a subspace of dimension lower than p. If so, the routine will give a warning, giving back the dimension of the subspace together with a direction describing a hyperplane containing this subspace.

Value

The output consists of a list. Each element of the list contains the following elements for each value α specified in the argument `alpha`.

`depth` The depth of the depth contour of level α. For halfspace depth this is equal to $floor(\alpha n)/n$. For projection depth, skewness-adjusted projection depth and directional depth, this equals to α.

`vertices` The coordinates of the vertices of the depth contour.

`empty` Logical indicating whether the corresponding depth region is empty.
`FALSE` indicates the depth region is non-empty.
`TRUE` indicates the depth region is empty.

dithered	Logical indicating whether dithering has been applied in the exact bivariate algorithm based on halfspace depth. FALSE indicates no dithering has been applied. TRUE indicates dithering has been applied.
converged	A vector of length m containing a flag indicating for each direction whether convergence was reached by the bisecting algorithm within the allowed max.iter number of steps.
type	Same as input parameter type.
dimension	If the data are lying in a lower dimensional subspace, the dimension of this subspace.
hyperplane	If the data are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert based on Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf

References

Ruts I., Rousseeuw P.J. (1996). Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, **23**, 153–168.

See Also

plotContours, bagdistance

Examples

```
# Compute and plot some halfspace depth contours of a two-dimensional dataset.
# The returned object is a ggplot2 object that may be edited
# using standard ggplot2 commands.

# One may consider different depth functions such as projection depth
# by changing the input parameter 'type'.
# By default the halfspace depth is used.
data(bloodfat)
Result <- depthContour(x = bloodfat,
                       alpha = c(0.03, 0.125, 0.25))
plotContours(x = bloodfat, depthContour = Result)


# Other options are projection depth, skewness-adjusted projection depth
# and directional projection depth
# they can be used by specifying type to be
# "projdepth", "sprojdepth" or "dprojdepth" respectively.
# When there is skewness in the data projection depth
# is less appropriate.

Result <- depthContour(x = bloodfat,
                       alpha = c(0.25, 0.35, 0.45),
```

```
                        type = "projdepth")
plotContours(x = bloodfat, depthContour = Result)

# The skewness-adjusted projection depth and directional projection depth
# better reflect the skewness in the data.
Result <- depthContour(x = bloodfat,
                        alpha = c(0.35, 0.45, 0.55),
                        type = "sprojdepth")
plotContours(x = bloodfat, depthContour = Result)

Result <- depthContour(x = bloodfat,
                        alpha = c(0.25, 0.35, 0.45),
                        type = "dprojdepth")
plotContours(x = bloodfat, depthContour = Result)
```

`dirOutl` *Directional outlyingness of points relative to a dataset*

Description

Computes the directional outlyingness of p-dimensional points `z` relative to a p-dimensional dataset `x`. For each multivariate point z_i, its directional outlyingness relative to `x` is defined as its maximal univariate directional outlyingness measured over all directions. To obtain the univariate directional outlyingness in the direction v, the dataset `x` is projected on v, and the robustly skew-adjusted standardized distance of $v'z_i$ to the median of the projected data points `x`v is computed. This is done through the estimation of 2 scales, one on each side of the median, using a 1-step M-estimator of scale.

Usage

```
dirOutl(x, z = NULL, options = list())
```

Arguments

`x`	An n by p data matrix.
`z`	An optional m by p matrix containing rowwise the points z_i for which to compute the directional outlyingness. If `z` is not specified, it is set equal to `x`.
`options`	A list of available options:

- `type`
  Determines the desired type of invariance and should be one of `"Affine"`, `"compWise"`. When the option `"Affine"` is used, the directions v are orthogonal to hyperplanes spanned by p observations from `x`. With the option `"compWise"`, the directional outlyingness is computed in the directions of the coordinate axes and combined through the Euclidean norm.
  Defaults to `"Affine"`.

- `ndir`
  When `type` is chosen to be `"Affine"`, determines the number of directions v by setting `ndir` to a specific number. Defaults to $250p$.
- `seed`
  A strictly positive integer specifying the seed to be used to select the directions.
  Defaults to 10.

Details

The directional outlyingness (DO) of multivariate data was introduced in Rousseeuw et al. (2018). It extends the Stahel-Donoho outlyingness towards skewed distributions.

Depending on the dimension p, different approximate algorithms are implemented. The affine invariant algorithm can only be used when $n > p$. It draws `ndir` times at random p observations from `x` and considers the direction orthogonal to the hyperplane spanned by these p observations. At most p out of n directions can be considered. The orthogonal invariant version can be applied to high-dimensional data. It draws `ndir` times at random 2 observations from `x` and considers the direction through these two observations. Here, at most 2 out of n directions can be considered. Finally, the shift invariant version randomly draws `ndir` vectors from the unit sphere.

The resulting DO values are invariant to affine transformations, rotations and shifts respectively provided that the `seed` is kept fixed at different runs of the algorithm. Note that the DO values are guaranteed to increase when more directions are considered provided the seed is kept fixed, as this ensures that the random directions are generated in a fixed order.

An observation from `x` and `z` is flagged as an outlier if its DO exceeds a cutoff value. This cutoff value is determined using the procedure in Rousseeuw et al. (2018). First, the logarithm of the DO values is taken to render their distribution more symmetric, after which a normal approximation yields a cutoff on these values. The cutoff is then transformed back by applying the exponential function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it. Furthermore, the univariate directional outlyingness of the projected points $\texttt{x}v$ is ill-defined when the scale in its denominator becomes zero. This can happen when many observations collapse. In these cases the algorithm will stop and give a warning. The returned values then include the direction v as well as an indicator specifying which of the observations of `x` belong to the hyperplane orthogonal to v.

Value

A list with components:

`outlyingnessX`	Vector of length n giving the directional outlyingness of the observations in `x`.
`outlyingnessZ`	Vector of length m giving the directional outlyingness of the points in `z` relative to `x`.
`cutoff`	Points whose directional outlyingness exceeds this cutoff can be considered as outliers with respect to `x`.
`flagX`	Observations of `x` whose directional outlyingness exceeds the cutoff receive a flag `FALSE`, regular observations receive a flag `TRUE`.

`flagZ` Points of `z` whose directional outlyingness exceeds the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.

`singularSubsets`
When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p-1$. In such a case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position.

`dimension` When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.

`hyperplane` When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust skew-adjusted scale of `x`v is equal to zero, this equals v.

`inSubspace` When a direction v is found such that DO(`x`v) is ill-defined, the observations from `x` which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

J. Raymaekers and P. Rousseeuw

References

Rousseeuw P.J., Raymaekers J., Hubert M. (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

`dprojdepth`, `dprojmedian`, `adjOutl`, `outlyingness`

Examples

```
# Compute the directional outlyingness of a two-dimensional dataset.
# Outliers are plotted in red.
data(geological)
BivData <- geological[c("MnO", "MgO")]
Result <- dirOutl(x = BivData)
IndOutliers <- which(!Result$flagX)
plot(BivData, pch = 16, col = "grey60")
points(BivData[IndOutliers, ], pch = 16, col = "red")

# The number of directions may be specified through
# the option list. The resulting directional outlyingness
# is monotone increasing in the number of directions.
Result1 <- dirOutl(x = BivData, options = list(ndir = 50))
Result2 <- dirOutl(x = BivData, options = list(ndir = 100))
which(Result2$outlyingnessX - Result1$outlyingnessX < 0)
# This is however not the case when the seed is changed
Result1 <- dirOutl(x = BivData, options = list(ndir = 50))
Result2 <- dirOutl(x = BivData, options = list(ndir = 100, seed = 950))
```

```
plot(Result2$outlyingnessX - Result1$outlyingnessX,
     xlab = "Index", ylab = "Difference in DO")
```

---

distSpace *distSpace*

Description

Computation of distance space representation.

Usage

```
distSpace(trainingData, testData = NULL, type = "bagdistance", options = NULL)
```

Arguments

trainingData	A list of n by p matrices containing multivariate data or a list of t by n by p arrays containing functional data. Each element of the list contains the observations from one group. The training data should contain at least two groups.
testData	An m by p matrix containing multivariate test data or a t by m by p array for functional data.
type	The distance used in the computations. For multivariate data one of the following options: bagdistance, outlyingness, adjOutl or dirOutl. Defaults to bagdistance. For functional data one of the following options: fbd, fSDO, fAO or fDO. Defaults to fbd.
options	A list of options to pass to the function computing the underlying distance. See bagdistance, outlyingness, adjOutl, dirOutl or fOutl for more information.

Details

The distance space representation is a tool in supervised classification and was introduced in Hubert et al. (2016) as a generalisation of the depth-depth representation of a multivariate sample. Based on a distance transform, an observation (be it multivariate or functional) is mapped to its representation in distance space. The distance transformation consists of mapping the observation to a vector containing at coordinate i the distance to the training group i. After transformation, any multivariate classifier may be used to classify new observations in distance space. Typically the k-nearest neighbour algorithm is used.

Different options are available to compute the distance to each of the training groups. For multivariate data, the user may choose between the bagdistance or any of the projection type distances including the Stahel-Donoho outlyingness, the adjusted outlyingness or the directional outlyingness. For functional data, the user may opt to employ the functional bagdistance (fbd), the functional Stahel-Donoho outlyingness (fSDO), the functional skweness-adjusted outlyingness (fAO) or the functional directional outlyingness (fDO). Options available in each of the underlying distance routines may be passed down using the options argument.

Value

A q by $(p + 1)$ matrix composed of two blocks. The first block contains the observations in the training set (rows) with in each column the distance to each of the groups. The last column contains a label indicating the original group membership of the observation. The second block contains the observations in the test set, if any, with in each column the distance to the different training groups. The last column contains an indicator signaling the observation was part of the test set.

Author(s)

P. Segaert

References

Hubert M., Rousseeuw P.J., Segaert P. (2017). Multivariate and functional classification using depth and distance. *Advances in Data Analysis and Classification*, **11**, 445–466.

Examples

```
data(plane)

# Build the training data
Mirage <- plane$plane1[, 1:25, 1, drop = FALSE]
Eurofighter <- plane$plane3[, 1:25, 1, drop = FALSE]
trainingData <- list(group1 = Mirage,
                     group2 = Eurofighter)

# Build the test data
Mirage.t <- plane$plane1[, 26:30, 1, drop = FALSE]
Eurofighter.t <- plane$plane3[, 26:30, 1, drop = FALSE]
testData <- abind::abind(Mirage.t, Eurofighter.t, along = 2)

# Transform the data into distSpace
Result <- distSpace(trainingData = trainingData, testData = testData, type="fbd")

# Plot the results
plotColors <- c(rep("orange", dim(Mirage)[2]),
                rep("blue", dim(Eurofighter)[2]),
                rep("green3", dim(testData)[2]))
plot(Result[, 1:2, ],
     col = plotColors, pch=16,
     xlab = "distance to Mirage", ylab = "distance to Eurofighter",
     main = "distSpace representation of Mirage and Eurofighter")
     legend("bottomleft", legend = c("Mirage","Eurofighter", "test data"), pch = 16,
       col = c("orange","blue", "green3"))
```

dprojdepth *Directional projection depth of points relative to a dataset*

Description

Computes the directional projection depth of p-dimensional points z relative to a p-dimensional dataset x.

Usage

```
dprojdepth(x, z = NULL, options = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the directional projection depth. If z is not specified, it is set equal to x.
options	A list of options to pass to the underlying dirOutl routine. See dirOutl for the full list of options.

Details

Directional projection depth is based on the directional outlyingness and is computed as $1/(1 + DO)$. As directional outlyingness extends the Stahel-Donoho outlyingness towards skewed distributions, the directional projection depth is suited for both elliptical distributions and skewed multivariate data.

It is first checked whether the data is found to lie in a subspace of dimension lower than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

See dirOutl for more details on the computation of the DO. To visualize the depth of bivariate data one can apply the mrainbowplot function. It plots the data colored according to their depth.

The output values of this function are based on the output of the dirOutl function. More details can be found there.

Value

A list with components:

depthX	Vector of length n giving the directional projection depth of the observations in x.
depthZ	Vector of length m giving the directional projection depth of the points in z.
cutoff	Points whose directional projection depth is smaller than this cutoff can be considered as outliers.
flagX	Observations of x whose directional outlyingness exceeds the cutoff receive a flag FALSE, regular observations receive a flag TRUE.

`flagZ` Points of `z` whose directional outlyingness exceeds the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.

`singularSubsets`
When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p-1$. In such a case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position.

`dimension` When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.

`hyperplane` When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust directional scale of xv is equal to zero, this equals v.

`inSubspace` When a direction v is found such that DO(xv) is ill-defined, the observations from `x` which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

J. Raymaekers

References

Rousseeuw, P.J., Raymaekers, J., Hubert, M. (2018). A measure of directional outlyingness with applications to image Data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

`dirOutl`, `dprojmedian`, `mrainbowplot`, `adjOutl`, `outlyingness`

Examples

```
# Compute the directional projection depth
# of a simple two-dimensional dataset.
# Outliers are plotted in red.

data(bloodfat)
Result <- dprojdepth(x = bloodfat)
IndOutliers <- which(!Result$flagX)
plot(bloodfat)
points(bloodfat[IndOutliers,], col = "red")

# A multivariate rainbowplot may be obtained using mrainbowplot.
plot.options = list(legend.title = "DPD")
mrainbowplot(x = bloodfat,
             depths = Result$depthX, plot.options = plot.options)

# Options for the underlying outlyingness routine may be passed
# using the options argument.
Result <- dprojdepth(x = bloodfat,
```

```
options = list(type = "Affine", ndir=100))
```

---

dprojmedian *Location estimates based on directional projection depth*

Description

Computes a directional projection depth based location estimate of a p-dimensional dataset x.

Usage

```
dprojmedian(x, dprojection.depths = NULL, options = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
dprojection.depths	Vector containing the directional projection depth of the observations in x.
options	A list of options to pass to the dprojdepth routine. See dprojdepth for more details.

Details

The algorithm depends on the function dprojdepth to compute the directional projection depth of the observations in x. If these depth values have already been computed they can be passed as an optional argument to save computing time. If not, directional projection depth values will be computed and the user can pass a list with options to the dprojdepth function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with component:

max	The observation of x with maximal directional projection depth. If multiple points have maximum depth, their center of gravity is returned.

Author(s)

J. Raymaekers

References

Rousseeuw P.J., Raymaekers J., Hubert M. (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

dirOutl, dprojdepth, adjOutl, outlyingness

Examples

```
# Compute a location estimate of a two-dimensional dataset.
data(bloodfat)
result <- dprojmedian(x = bloodfat)
plot(bloodfat, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)

# Options for the underlying dprojdepth routine may be passed
# using the options argument.
result <- dprojmedian(x = bloodfat, options = list(type = "Rotation", ndir = 100))
plot(bloodfat, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)

# One may also compute the depth values of the observations in the data
# separately. This avoids having to recompute them when conmputing the median.
depth.result <- dprojdepth(x = bloodfat)
result <- dprojmedian(x = bloodfat, dprojection.depths = depth.result$depthX)
```

---

fHeatmap *Draws a heatmap of functional depth values or distances*

Description

Draws a heatmap of depth values or distances of functional data.

Usage

```
fHeatmap(rowValues, cellValues, type, legend.title = "")
```

Arguments

rowValues	Vector of functional depth or distance values. Each value should correspond with the functional depth or distance of an observation from a (multivariate) functional data set.
cellValues	Matrix of multivariate depth or distance values. The value in row i and column j should correspond with the multivariate depth or distance of observation i at time point j.
type	One of "depth" or "distance". Determines whether a depth or a distance heatmap is drawn.
legend.title	Title of the legend.

Details

On the vertical axis the functional data are sorted from top to bottom according to their functional depth or distance value as provided in `rowValues`. When `type = "depth"`, the deepest observation is put at the bottom. When `type = "distance"`, the rows are sorted in decreasing order of their functional distance.

Each cell of the map is colored according to the multivariate depth provided in `cellValues`. For a depth-based heatmap, the smallest depth value is white and the overall highest depth value is colored dark green. A distance-based heatmap colors the cells from white to dark red.

Author(s)

P. Segaert

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Examples

```
data(octane)
Result <- mfd(octane, diagnostic = TRUE, type = "sprojdepth")
Plot <- fHeatmap(rowValues = Result$MFDdepthZ,
                 cellValues = Result$crossdepthZ,
                 type = "depth",
                 legend.title = "SPD")
Plot

Result <- fOutl(octane, diagnostic = TRUE, type = "fAO")
Plot <- fHeatmap(rowValues = Result$fOutlyingnessZ,
                 cellValues = Result$crossDistsZ,
                 type = "distance",
                 legend.title = "AO")
Plot
```

---

fom *Draws the Functional Outlier Map (FOM)*

Description

Creates the Functional Outlier Map, a graphical tool to detect outliers in multivariate functional data. Depending on the position of a multivariate curve in the FOM, different types of outliers may be distinguished.

Usage

```
fom(fOutlResult, cutoff = FALSE)
```

Arguments

`fOutlResult`	The return of a call to `fOutl`
`cutoff`	Boolean indicating whether the cutoff should be drawn.

Details

The fom is only applicable when `fOutl` is called with the following options: `diagnostic = TRUE` and `type` equaling either `fAO` or `fDO`.

The functional outlier map was proposed by Hubert et al. (2015) and subsequently improved by Rousseeuw et al. (2018). It consists of a graphical tool to detect outliers in multivariate functional data based on functional outlyingness measures such as the fAO or fDO (see `fOutl`).

The coordinates of the points in the FOM correspond to two outlyingness indicators. On the horizontal axis, the functional outlyingness as obtained by the routine `fOutl` is plotted. On the vertical axis the scaled standard deviation of the cross-sectional outlyingness measures across time are plotted. The FOM thus consists of the following points representing the curve i: $(fOutl_i; std_j(fOutl_i(t_j))/(1+fOutl_i))$. The scaling of the standard deviation is added to ensure that curves with a different location, measured from the center, but with the same relative variability have a similar y-coordinate.

For some underlying multivariate outlyingness measures, the output of `fOutl` contains an additional argument signaling whether curve i is outlying as measured in the multivariate space at time point j. This information is incorporated in the FOM by plotting the points with a different degree of outlyingness by a different symbol. Curves that are outlying in the multivariate space determined by at least 75% of the total number of time points in the domain are plotted using filled diamonds. Similarly curves that are outlying in at least 50% or 25% of the time points are plotted in filled squares and filled triangles respectively. Curves that are outlying in fewer than 25% of the time points are plotted using filled circles.

The user may opt to draw a cutoff curve for the detection of outliers on the FOM. This cutoff was introduced in Rousseeuw et al. (2018) and is based on the euclidean distance between the points on the FOM and the origin, after scaling with the median. More specifically, let $(fOutl_i; vOutl_i) = (fOutl_i; std_j(fOutl_i(t_j))/(1+fOutl_i))$ and let $cOutl_i^2 = (fOutl_i/median(fOutl))^2+(vOutl_i/median(vOutl))^2$. Finally, with $lcOutl_i = \log(0.1+coutl_i)$,an observation lies outside of the cutoff when $(lcOutl_i - median(lcOutl))/mad(lcOutl) > \Phi^{-1}(0.995)$.

This FOM may be read in a way similar to the outlier map of robust regression (Rousseeuw and van Zomeren 1990) and the outlier map of robust principal components (Hubert et al. 2005). Points in the lower left part of the FOM represent regular curves which hold a central position in the dataset. Points in the lower right part are curves with a high $fOutl$ but a low variability of their cross-sectional $fOutl$ values. This happens for shift outliers, i.e. curves which have the same shape as the majority but are shifted on the whole domain. Points in the upper left part have a low $fOutl$ but a high variability in cross-sectional $fOutl$. Typical examples are isolated outliers, i.e. cuves which only display outlyingness over a small part of their domain. The points in the upper right part of the FOM have both a high $fOutl$ and a high cross-sectional $fOutl$. These correspond to curves which are strongly outlying on a substantial part of their domain.

Author(s)

P. Segaert

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Rousseeuw P.J., Raymaekers J., Hubert M., (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

fOutl

Examples

```
data(octane)

# To construct the FOM, one first need to compute
# the functional outlyingness.
# Note that the option diagnostic in fOutl must be
# set to TRUE. If not calling fom will result in an
# error
Result <- fOutl(octane, alpha = 0, type = "fAO", diagnostic = TRUE)
fom(Result)

# The user may opt to draw a cut off line seperating the outliers.
# which will be plotted in red
fom(Result, cutoff = TRUE)

# Six observations are flagged as outliers. These correspond to
# the samples with added ethanol.
```

---

fOutl *Functional outlyingness measures for functional data*

Description

Computes several measures of functional outlyingness for multivariate functional data.

Usage

```
fOutl(x, z = NULL, type = "fAO", alpha = 0, time = NULL,
      diagnostic = FALSE, distOptions = NULL)
```

Arguments

x	A three dimensional t by n by p array, with t the number of observed time points, n the number of functional observations and p the number of measurements for every functional observation at every time point.

z	An optional three-dimensional t by m by p array, containing the observations for which to compute the functional outlyingness with respect to x. If z is not specified, it is set equal to x. The time points of z should correspond to those of x.
type	The outlyingness measure used in the computations. One of the following options: "fAO", "fSDO", "fDO" or "fbd". Defaults to "fAO".
alpha	Specifies the weights at every cross-section. When alpha = 0, uniform weights are used. Otherwise alpha should be a weight vector of length t. Defaults to 0.
time	If the measurements are not equidistant, a sorted numeric vector containing a set of time points. Defaults to 1:t.
diagnostic	If set to TRUE, the output contains some additional components: crossDists: an n by t matrix containing the multivariate outlyingness of each observation at each time point locOutl: output containing flags for local outlyingness (see "Value" for more details) Defaults to FALSE.
distOptions	A list of options to pass to the function computing the cross-sectional distances. See adjOutl, outlyingness, dirOutl, or bagdistance.

Details

The functional outlyingness of a multivariate curve with respect to a given set of multivariate curves is defined as the weighted average of its multivariate outlyingness at each time point (Hubert et al., 2015). The functional outlyingness can be computed in all dimensions p using the adjusted outlyingness (fAO), the directional outlyingness (fDO), the Stahel-Donoho outlyingness (fSDO) or the bagdistance (fbd).

When the data array z is specified, the functional outlyingness and diagnostic information for the data array x is also returned whenever the underlying outlyingness routine allows it. For more information see the specific routines listed in the section "See Also".

In some situations, additional diagnostics are available to flag outlying time points. At each time point, observations from the data array x are marked if they are flagged as outliers. The observations from the data array x are marked if their scaled outlyingness is larger than a prescribed cutoff value from the chi-square distribution. For more details see the respective outlyingness routines.

It is possible that at certain time points a part of the algorithm can not be executed due to e.g. exact fits. In that case the weight of that particular time point is set to zero. A warning is issued at the end of the algorithm to signal these time points. Furthermore the output contains an extra argument giving the indices of the time points where problems occured.

Value

A list with the following components:

fOutlyingnessX	Vector of length n containing the functional outlyingness of every curve from x.

`fOutlyingnessZ` Vector of length m containing the functional outlyingness of every curve from `z`.

`weights` Vector of weights according to the input parameter `alpha`.

`crossDistsX` An n by t matrix containing the multivariate outlyingness of each observation of `x` at each point. Only provided if the input parameter `diagnostic` is set to `TRUE`.

`crossDistsZ` An m by t matrix containing the multivariate outlyingness of each observation of `z` at each point. Only provided if the input parameter `diagnostic` is set to `TRUE`.

`locOutlX` An n by t matrix flagging local outlyingness for `x`. Only provided if the input parameter `diagnostic` is set to TRUE.
The (i, j)th element takes value 1 if curve x_i is outlying at time point j.

`locOutlZ` An m by t matrix flagging local outlyingness for `z`. Only provided if the input parameter `diagnostic` is set to TRUE.
The (i, j)th element takes value 1 if curve z_i is outlying at time point j.

`IndFlagExactFit`
Vector containing the indices of the time points for which an exact fit is detected.

Author(s)

P. Segaert

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods and Applications,* **24**, 177–202.

Hubert M., Rousseeuw P.J., Segaert P. (2017). Multivariate and functional classification using depth and distance. *Advances in Data Analysis and Classification*, **11**, 445–466.

See Also

`bagdistance`, `outlyingness`, `adjOutl`, `dirOutl`, `fom`

Examples

```
data(octane)
Data <- octane

# When the option diagnostic is set to TRUE, a crude diagnostic
# to detect outliers can be extracted from the local outlyingness
# indicators.
Result <- fOutl(x = Data, type = "fAO", diagnostic = TRUE)
matplot(Data[,,1], type = "l", col = "black", lty = 1)
for (i in 1:dim(Data)[2]) {
  if(sum(Result$locOutlZ[i, ]) > 0) {
    obsData <- matrix(Data[,i,1], nrow = 1)
    obsData[!Result$locOutlZ[i,]] <- NA
    obsData <- rbind(obsData, obsData)
```

```
    matpoints(t(obsData), col = "red", pch = 15)
  }
}
# For more advanced outlier detection techniques, see the
# fom routine.
```

`geological` *Composition of elements in soil samples around the Baltic Sea*

Description

This data originates from a geological survey on the composition in agricultural soils from 10 countries surrounding the Baltic Sea. Top soil (0-25 cm) and bottom soil (50-75 cm) samples from 768 sites were analysed. This data frame contains the measurements corresponding to the total concentration of four elements in the top soil samples.

Usage

```
data(geological)
```

Format

A data frame containing the following variables:

`Fe2O3` Iron Oxide

`MgO` Magnesium oxide

`MnO` Manganese oxide

`TiO2` Titanium dioxide

Source

Reimann C., Siewers U., Tarvainen T., Bityukova L., Eriksson J., Gilucis A., Gregorauskiene V., Lukashev V., Matinian N.N., Pasieczna A. (2000). Baltic soil survey: total concentrations of major and selected trace elements in arable soils from 10 countries around the Baltic Sea. *Science of the Total Environment*, **257**, 155–170.

References

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Examples

```
data(geological)
plot(geological)
```

glass *EXPMA spectra of glass samples*

Description

The glass data set studied by Lemberge et al. (2000) consists of 180 different 16-17th century archeological glass samples. Electron Probe X-ray Microanalysis (EPXMA) intensities across 750 energy levels are recorded using a Jeol JSM 6300 scanning electron microscope equipped with an energy-dispersive Si(Li) X-ray detection system (SEM-EDX).

Usage

```
data(glass)
```

Format

A three-dimensional $t = 750$ by $n = 180$ by $p = 1$ array.

Source

Lemberge P., De Raedt I., Janssens K.H., Wei F., Van Espen P.J. (2000). Quantitative Z-analysis of 16th-17th century archaeological glass vessels using PLS regression of EPXMA and μ-XRF data. *Journal of Chemometrics*, **14**, 751–763.

References

Hubert M., Rousseeuw P.J., Vanden Branden K. (2005). ROBPCA: a new approach to robust principal component analysis. *Technometrics*, **47**, 64–79.

Examples

```
data(glass)
matplot(glass[,,1], type = "l", lty = 1, col = "black")
```

hdepth *Halfspace depth of points relative to a dataset*

Description

Computes the halfspace depth of p-dimensional points z relative to a p-dimensional dataset x. Computation is exact for $p \leq 3$ and approximate when $p > 3$. For the approximate algorithm the halfspace depth is computed as the minimal univariate halfspace depth over many directions. To obtain the univariate halfspace depth in the direction v, the dataset x is projected on v, and the univariate location depth of the points of $v'z_i$ to xv is computed.

Usage

```
hdepth(x, z = NULL, options = list())
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the halfspace depth. If z is not specified, it is set equal to x.
options	A list of available options:

- type
  Determines the desired type of invariance for the approximate algorithm and should be one of "Affine", "Rotation" or "Shift". When the option "Affine" is used, the directions v are orthogonal to hyperplanes spanned by p observations from x. When the option "Rotation" is used, the directions pass by two randomly selected observations from x. With the option "Shift", directions are randomly generated.
  Defaults to "Affine".
- ndir
  Determines the number of directions v by setting ndir to a specific number or to "all". In the latter case, an exhaustive search over all possible directions (according to type) is performed. If ndir is larger than the number of possible directions, the algorithm will automatically use this setting.
  Defaults to $250p$ when type="Affine", to 5000 when type="Rotation" and to 12500 when type="Shift".
- approx
  The user may force approximate calculation in two or three dimensions by setting this option to TRUE.
  Defaults to FALSE.
- seed
  A strictly positive integer specifying the seed to be used by the C++ code.
  Defaults to 10.

Details

Halfspace depth has been introduced by Tukey (1975). The halfspace depth of a point z_i is defined as the minimal number of observations from x that are contained in any closed halfspace with boundary through z_i.

In dimensions $p = 2$ and $p = 3$ the computations are by default carried out exactly using the algorithms described in Rousseeuw and Ruts (1996) and Rousseeuw and Struyf (1998). This yields an affine invariant measure of depth. Approximate algorithms are also implemented which are affine, rotation or shift invariant, depending on the value chosen for type. They can be used in any dimension. The shift invariant algorithm coincides with the random Tukey depth (Cuesta-Albertos and Nieto-Reyes, 2008).

The resulting halfspace depth values are invariant to affine transformations when the exact algorithm is used and invariant to affine transformations, rotations and shifts depending on the choice for type, provided that the seed is kept fixed at different runs of the algorithm. Note that the halfspace depth

values values are guaranteed to decrease when more directions are considered, provided the seed is kept fixed, as this ensures that the random directions are generated in a fixed order.

If the halfspace depth needs to be computed for m points z_i, it is recommended to apply the function once with the matrix z as input, instead of applying it m times with input vectors z_i, as numerous computations can be saved. The approximate algorithms automatically then also compute the depth values of the observations in x. When only the halfspace depth of the observations in x is required, the call to the function should be `hdepth(x)` or equivalently `hdepth(x,x)`. In that case the depth values will be stored in the 'depthZ' output field. For bivariate data these will be the exact values by default.

To visualize the depth of bivariate data one can apply the `mrainbowplot` function. It plots the data colored according to their depth.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with components:

`depthX` Vector of length n giving the halfspace depth of the observations in x. By default exact if $p \leq 3$ and approximate if $p > 3$ or the option `approx` is set to `TRUE`.

`depthZ` Vector of length m giving the halfspace depth of the points in z relative to x. By default exact if $p \leq 3$ and approximate if $p > 3$ or the option `approx` is set to `TRUE`.

`singularSubsets`
When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p-1$. In that case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position. When the input parameter type is equal to `"Rotation"` it is possible that two randomly selected points of the data coincide due to ties in the data. In this case this value signals how many times this happens.

`dimension` When the data x are lying in a lower dimensional subspace, the dimension of this subspace.

`hyperplane` When the data x are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert based on Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf, and `C++` code by P. Segaert and K. Vakili.

References

Tukey J. (1975). Mathematics and the picturing of data. *Proceedings of the International Congress of Mathematicians,* **2**, 523–531, Vancouver.

Rousseeuw P.J., Ruts I. (1996). AS 307: Bivariate location depth. *Journal of the Royal Statistical Society: Series C*, **45**, 516–526.

Rousseeuw P.J., Struyf A. (1998). Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, **8**, 193–203.

Cuesta-Albertos J., Nieto-Reyes A. (2008). The random Tukey depth. *Computational Statistics & Data Analysis*, **52**, 4979–4988.

See Also

hdepthmedian, mrainbowplot, bagdistance, bagplot

Examples

```
# Compute the halfspace depth of a simple
# two-dimensional dataset.
data(cardata90)
Result <- hdepth(x = cardata90)
mrainbowplot(cardata90, depths = Result$depthZ)

# In two dimensions we may also opt to use the
# approximate algorithm. The number of directions
# may be specified through the option list.
options <- list(type = "Rotation",
                ndir = 750,
                approx = TRUE)
Result <- hdepth(x = cardata90, options = options)
# The resulting halfspace depth is monotone decreasing
# in the number of directions.
options <- list(type = "Rotation",
                ndir = 10,
                approx = TRUE)
Result1 <- hdepth(x = cardata90, options = options)
options <- list(type = "Rotation",
                ndir = 500,
                approx = TRUE)
Result2 <- hdepth(x = cardata90, options = options)
which(Result1$depthZ - Result2$depthZ < 0)
# This is however not the case when the seed is changed
options <- list(type = "Rotation",
                ndir = 10,
                approx = TRUE)
Result1 <- hdepth(x = cardata90, options = options)
options <- list(type = "Rotation",
                ndir = 50,
                approx = TRUE,
                seed = 897)
Result2 <- hdepth(x = cardata90, options = options)
which(Result1$depthZ - Result2$depthZ < 0)
plot(Result1$depthZ - Result2$depthZ,
     xlab = "Index", ylab = "Difference in halfspace depth")

# We can also consider directions through two data
# points. If the sample is small enough one may opt
# to search over all choose(n,2) directions.
```

```
# Note that the computational load increases substantially
# as n becomes larger.
options <- list(type = "Rotation",
                ndir = "all",
                approx = TRUE)
Result1 <- hdepth(x = cardata90, options = options)

# Alternatively one may consider randomly generated directions.
options <- list(type = "Shift",
                ndir = 250,
                approx = TRUE)
Result1 <- hdepth(x = cardata90, options = options)
```

`hdepthmedian` *Location estimates based on halfspace depth.*

Description

Computes the halfspace median and its corresponding halfspace depth for a p-dimensional data set `x`. Computation is exact for $p \leq 2$ and approximate for $p > 2$.

Usage

```
hdepthmedian(x, maxdir = NULL)
```

Arguments

`x`	An n by p data matrix.
`maxdir`	The number of projections used in the approximate algorithm. Defaults to $250p$.

Details

The halfspace median, or Tukey median, is the multivariate point with largest halfspace depth with respect to the data `x`. This point is not always unique. In that case the halfspace median corresponds to the center of gravity of the convex set of deepest points.

It is first checked whether the data is found to lie in a subspace of dimension lower than p. If so, the routine will give a warning, giving back the dimension of the subspace together with a direction describing a hyperplane containing this subspace.

For bivariate data the exact algorithm of Rousseeuw and Ruts (1998) is applied. When the data are not in general position (i.e. when there is a line containing more than two observations) dithering is performed by adding random Gaussian noise to the data. In this case the output argument `dithered` will contain a flag.

When $p > 2$ the approximate algorithm of Struyf and Rousseeuw (2000) is applied. It is an iterative procedure based on projections. Their number can be chosen by the input parameter `maxdir`.

Value

A list containing:

`median`	The coordinates of the halfspace median. Approximate when $p > 2$.
`depth`	The halfspace depth of the halfspace median. Approximate when $p > 2$.
`dithered`	Logical indicating whether dithering has been applied in the exact algorithm. `FALSE` indicates no dithering has been applied. `TRUE` indicates dithering has been applied.
`ndir`	The number of projections used by the approximate algorithm. Due to the possibility of singularity of certain p subsamples it is possible that not all `maxdir` directions are evaluated.
`AlgStopFlag`	Indicates which stopping rule is used by the approximate algorithm. 0 indicates the maximum number of projections was reached 1 indicates no improvement of the location estimate was made after $10(p + 1)$ steps.
`dimension`	If the data are lying in a lower dimensional subspace, the dimension of this subspace.
`hyperplane`	If the data are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert based on Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf

References

Rousseeuw P.J., Ruts I. (1998). Constructing the bivariate Tukey median. *Statistica Sinica*, **8**, 827–839.

Struyf A., Rousseeuw P.J. (2000). High-dimensional computation of the deepest location. *Computational Statistics & Data Analysis*, **34**, 415–436.

Examples

```
# Compute a location estimate of a two-dimensional dataset.
data(cardata90)
Result <- hdepthmedian(x = cardata90)
plot(cardata90, pch = 16)
points(Result$median, col = "red", pch = 18, cex = 1.5)
```

`medcouple` *A robust measure of skewness for univariate data*

Description

Computes the medcouple, a robust measure of skewness for univariate data. For multivariate data the medcouple is computed on each column of the data matrix.

Usage

```
medcouple(x, do.reflect = NULL)
```

Arguments

`x`	An n by p data matrix.
`do.reflect`	Logical indicating whether the medcouple should also be computed on the reflected sample `-x`, with final result $(mc(\texttt{x}) - mc(-\texttt{x}))/2$. Defaults `TRUE` when $n <= 100$ and to `FALSE` otherwise.

Details

The medcouple is a robust measure of skewness yielding values between -1 and 1. For left- and right-skewed data the medcouple is negative and positive respectively.

The medcouple is defined as the median of the kernel function $h(x_i, x_j) = \frac{(x_j - med(x)) - (med(x) - x_i)}{x_j - x_i}$ evaluated over all couples (x_i, x_j) where x_i is smaller than the median of `x` and x_j larger than the median of `x`. When there are multiple observations tied to the median, the kernel is defined separately as the denominator is not defined for these observations. Let $m_1 < ... < m_k$ denote the indices of the observations which are tied to the median. Then $h(x_{m_i}, x_{m_j})$ is defined to equal -1 if $i + j - 1 < k$, 0 when $i + j - 1 = k$ and $+1$ if $i + j - 1 > k$. To compute the medcouple an algorithm with time complexity $O(nlog(n))$ is applied. For details, see https://en.wikipedia.org/wiki/Medcouple.

For numerical accuracy it is advised, for small data sets, to compute the medcouple on both `x` and `-x`. The final value of the medcouple may then be obtained as a linear combination of both calculations. This procedure is warranted by the properties of the medcouple. Indeed the medcouple of the distribution X equals minus the medcouple of the reflected distribution $-X$. Moreover the medcouple is location and scale invariant.

Note that missing values are not allowed.

Value

`mc`	A p-vector containing the medcouple of each column of the data matrix `x`.

Author(s)

P. Segaert with original code from M. Maechler and G. Brys.

References

Brys G., Hubert M., Struyf A. (2004). A robust measure of skewness. *Journal of Computational and Graphical Statistics*, **13**, 996–1017.

Examples

```
# Calculate the medcouple of a bivariate data set.
# Note that the medcouple of each variable is returned.
data(bloodfat)
medcouple(bloodfat)

# For smaller data sets it is advisable to compute
# the medcouple on both the sample and the reflected sample.
small.data <- bloodfat[1:25,]
medcouple(small.data, do.reflect = FALSE)
-medcouple(-small.data, do.reflect = FALSE)
# Small difference are due to numerical instabilities.
# Use the option do.reflect to increase expected accuracy.
medcouple(small.data, do.reflect = TRUE)
```

---

mfd *Multivariate functional depth for functional data*

Description

Computes the multivariate functional depth for multivariate functional data.

Usage

```
mfd(x, z = NULL, type = "hdepth", alpha = 0, time = NULL, diagnostic = FALSE,
    depthOptions = NULL)
```

Arguments

x	A three-dimensional t by n by p array, with t the number of observed time points, n the number of functional observations and p the number of measurements for every functional observation at every time point.
z	An optional three-dimensional t by m by p array, containing the observations for which to compute the multivariate functional depth with respect to x. If z is not specified, it is set equal to x. The time points of z should correspond to those of x.
type	The depth used in the computations. One of the following options: "hdepth", "projdepth", "sprojdepth", "dprojdepth", "sdepth". Defaults to "hdepth".

alpha	Specifies the weights at every cross-section. When alpha = 0, uniform weights are used. Weights following equation (2) in Claeskens et al. (2014) are obtained by setting alpha to a number smaller than the maximal depth at any time point. The weights are then proportional to the volume of the α-depth regions at each cross-section. Otherwise alpha should be a weight vector of length t. Defaults to 0.
time	If the measurements are not equidistant, a sorted numeric vector containing a set of time points. Defaults to 1:t.
diagnostic	If set to TRUE, the output contains some additional components: crossDepths: an n by t matrix containing the multivariate depth of each observation at each time point locOutl: output containing flags for local outlyingness (see "Value" for more details) Defaults to FALSE.
depthOptions	A list of options to pass to the function computing the cross-sectional depths. See hdepth, projdepth, sprojdepth, dprojdepth or sdepth.

Details

The multivariate functional depth of a multivariate curve with respect to a given set of multivariate curves is defined as the weighted average of its multivariate depth at each time point (Claeskens et al., 2014). The MFD can be computed in all dimensions p using halfspace depth, projection depth, skewness-adjusted projection depth and directional projection depth. For $p <= 2$ also simplicial depth is available.

When the data array z is specified, the MFD depth and diagnostic information for the data array x is also returned whenever the underlying depth routine allows it. For more information see the specific depth routines listed in the section "See Also".

For the weight vector, three options are available: uniform weights, user-defined weights or weights proportional to the volume of the α-depth contour at each time point. The α-depth contours are computed using the depthContour function.

In some situations, additional diagnostics are available to flag outlying time points, as described in Hubert et al. (2012). At each time point, observations from the data array x are marked if they are flagged as outliers. When using any of the projection depth measures, this flag is automatically returned by the corresponding functions. When using halfspace depth, the diagnostic is only available for bivariate curves. The observations from the data array x are marked if they are flagged as outliers by the bagplot, or similarly if their bagdistance is larger than 3 at that time point. This can be seen as a measure of local outlyingness. The option is not available for simplicial depth.

A heatmap of the cross-sectional depth values can be drawn by setting diagnostic to TRUE and passing the results to fHeatmap.

It is possible that at certain time points a part of the algorithm can not be executed due to e.g. exact fits. In that case the weight of that particular time point is set to zero. A warning is issued at the end of the algorithm to signal these time points. Furthermore the output contains an extra argument giving the indices of the time points where problems occured.

Value

A list with the following components:

`MFDdepthX`	Vector of length n containing the MFD depth of every curve from `x`.
`MFDdepthZ`	Vector of length m containing the MFD depth of every curve from `z`.
`weights`	Vector of weights according to the input parameter `alpha`.
`crossDepthsX`	An n by t matrix containing the multivariate depth of each observation of `x` at each time point. Only provided if the input parameter `diagnostic` is set to `TRUE`.
`crossDepthsZ`	An m by t matrix containing the multivariate depth of each observation of `z` at each time point. Only provided if the input parameter `diagnostic` is set to `TRUE`.
`locOutlX`	An n by t matrix flagging local outlyingness for `x`. Only provided if the input parameter `diagnostic` is set to TRUE. The (i, j)th element takes value 1 if curve x_i is outlying at time point j.
`locOutlZ`	An m by t matrix flagging local outlyingness for `z`. Only provided if the input parameter `diagnostic` is set to TRUE. The (i, j)th element takes value 1 if curve z_i is outlying at time point j.
`IndFlagExactFit`	Vector containing the indices of the time points for which an exact fit is detected.
`IndFlagBag`	Vector containing the indices of the time points for which the bagplot could not be computed.
`IndFlagIso`	Vector containing the indices of the time points for which the cross-sectional α-depth contours could not be computed.
`IndFlagAlpha`	Vector containing the indices of the time points for which the volume of the cross-sectional α-depth contours could not be computed.

Author(s)

P. Segaert and M. Hubert

References

Claeskens G., Hubert M., Slaets L., Vakili K. (2014). Multivariate functional halfspace depth. *Journal of the American Statistical Association,* **109**, 411–423.

Hubert M., Claeskens G., De Ketelaere B., Vakili K. (2012). A new depth-based approach for detecting outlying curves. In *Proceedings of COMPSTAT 2012, edited by A. Colubi, K. Fokianos, G. Gonzalez-Rodriguez, E.J. Kontoghiorghes,* 329–340.

See Also

`depthContour`, `hdepth`, `projdepth`, `sprojdepth`, `dprojdepth`, `sdepth`, `fHeatmap`

Examples

```
data(octane)
Result <- mfd(x = octane, alpha = 0.125, diagnostic = TRUE)

Plot <- fHeatmap(rowValues = Result$MFDdepthZ,
                 cellValues = Result$crossdepthZ,
                 type = "depth",
                 legend.title = "HD")
Plot
```

`mfdmedian` *Multivariate functional median for functional data*

Description

Computes the multivariate functional median, an estimate for the central tendency of multivariate functional data.

Usage

```
mfdmedian(x, type = "hdepth", crossDepthsX = NULL,
          depthOptions = NULL, centerOption = "maxdepth")
```

Arguments

`x`	A three-dimensional t by n by p array, with t the number of observed time points, n the number of functional observations and p the number of measurements for every functional observation at every time point.
`type`	The depth used in the computations. One of the following options: `"hdepth"`, `"projdepth"`, `"sprojdepth"`, `"dprojdepth"`, `"sdepth"`. Defaults to `"hdepth"`.
`crossDepthsX`	Depth values at each time point. Can be used to save computing time.
`depthOptions`	A list of options to pass to the function that computes the cross-sectional depths.
`centerOption`	When equal to `"maxdepth"` the functional median equals at each time point the point with cross-sectional maximal depth. When `type` is equal to `"projdepth"`, also a weighted center of gravity can be computed based on Huber weights (see `projmedian`). Then `centerOption` should be set to `"Huber"`. Defaults to `"maxdepth"`.

Details

The multivariate functional median of a multivariate functional data set is defined as the multivariate curve connecting the cross-sectional multivariate depth medians (Claeskens et al., 2014). The MFD median can be computed in all dimensions p using halfspace depth, projection depth, skewness-adjusted projection depth or directional projection depth. The simplicial depth can only be used for $p \leq 2$.

It is possible that at certain time points a part of the algorithm can not be executed due to e.g. exact fits. In that case the estimate for the center will be set to NaN. A warning is issued at the end of the algorithm to signal these time points. Furthermore the output contains an extra argument giving the indices of the time points where problems occured.

Value

A list with the following component:

`MFDmedian` An t by p matrix containing the estimated central curve.

`IndFlagExactFit`
Vector containing the indices of the time points for which an exact fit is detected.

Author(s)

P. Segaert, M. Hubert

References

Claeskens G., Hubert M., Slaets L., Vakili K. (2014). Multivariate functional halfspace depth. *Journal of the American Statistical Association,* **109**, 411–423.

See Also

hdepth, projdepth, sprojdepth, dprojdepth, sdepth, mfd

Examples

```
# Consider a bivariate functional sample.
data(characterA)
Data <- characterA[, 1:50, ]
Result <- mfdmedian(Data)

# Plot the data and the functional median
par(mfrow = c(1,2))
matplot(Data[, , 1], type = "l", col = "black", lty = 1, ylab = "x-coordinate")
matlines(Result$MFDmedian[, 1], type = "l", col = "red", lwd = 2)
matplot(Data[, , 2], type = "l", col = "black", lty = 1, ylab = "y-coordinate")
matlines(Result$MFDmedian[, 2], type = "l", col = "red", lwd = 2)
par(mfrow = c(1,1))

# Other depth functions such as the adjusted outlyingness may also
# be used to determine the cross-sectional depth median.
# In this case the depth median is computed by the
# sprojmedian routine.
# Optional arguments used by the sprojmedian routine may be specified
# using the depthOptions. For example one might choose the
# "Rotation" option with 300 directions.
Result <- mfdmedian(Data, type = "sprojdepth",
                    depthOptions = list(type = "Rotation",
                                        ndir = 300))
par(mfrow = c(1,2))
```

```
matplot(Data[, , 1], type = "l", col = "black", lty = 1, ylab = "x-coordinate")
matlines(Result$MFDmedian[, 1], type = "l", col = "red", lwd = 2)
matplot(Data[, , 2], type = "l", col = "black", lty = 1, ylab = "y-coordinate")
matlines(Result$MFDmedian[, 2], type = "l", col = "red", lwd = 2)
par(mfrow = c(1,1))

# If the user already placed a call to the mfd routine
# with the diagnostic options set to TRUE, the
# mfdmedian can easily be obtained by passing the cross-sectional
# depths. This considerably saves computing time.
tResult <- mfd(x = Data, type = "sprojdepth", diagnostic = TRUE)
Result <- mfdmedian(Data, type = "sprojdepth",
                    crossDepthsX = tResult$crossdepthX,
                    )

# Univariate curves should also be represented as arrays
Data.x <- Data[, , 1, drop = FALSE]
Result <- mfdmedian(Data.x)
matplot(Data.x[ , , 1], type = "l", col = "black", lty = 1, ylab = "x-coordinate")
matlines(Result$MFDmedian[, 1], type = "l", col = "red", lwd = 2)
```

`mrainbowplot` *Rainbow plot for bivariate data*

Description

Makes a scatterplot of bivariate data and colors the observations according to their depth value.

Usage

```
mrainbowplot(x, depths, col = NULL, plot.options = list())
```

Arguments

`x`	An n by 2 data matrix.
`depths`	A column vector of length n. The depth values of the observations in `x`. The coloring is based on these depth values.
`col`	An $m > 2$ by 3 matrix. Colors in rgb format. The user may use this argument to set the colorscale of the depth range. The first row should contain the rgb values for the lowest depth value, the last row the rgb values of the color for the deepest depth value. Colors for other depth values are interpolated. When more than two rows are provided the color range will be equidistantly divided over the different colors.
`plot.options`	A list of available options: • legend.title Title of the legend. Defaults to `"Depth"`. • point.size Numeric defining the size of the points in the plot. Defaults to 4.

Details

The plot is made using ggplot2. The plot itself is returned by the function and is fully customisable using standard ggplot2 commands. Similar plots for multivariate data with $p > 2$ can be made using the ggpairs function in the library `GGally`.

Author(s)

P. Segaert

Examples

```
data(cardata90)
Result <- projdepth(x = cardata90)
plot.options <- list(legend.title = "PD")
plot <- mrainbowplot(cardata90,
                     depths = Result$depthZ,
                     plot.options = plot.options)
plot + ggtitle("Rainbowplot based on projection depth")

# The default color range may be adjusted using the col argument.
RGBmatrix <- c(1, 0, 0, #Red
               1, 1, 1, #White
               0, 1, 0) #Green
RGBmatrix <- matrix(RGBmatrix, ncol = 3, byrow = TRUE)
plot <- mrainbowplot(cardata90,
                     depths = Result$depthZ,
                     col = RGBmatrix,
                     plot.options = plot.options)
plot + ggtitle("Rainbowplot based on projection depth")
```

---

mri *Intensities of MRI images*

Description

Felipe et al. (2005) obtained intensities of MRI images of 9 different parts of the human body (plus a group consisting of all remaining body regions, which was of course very heterogeneous). They then transformed their data to univariate curves.

Usage

```
data(mri)
```

Format

A list of arrays corresponding to each bodypart. For each bodypart, a three-dimensional $t = 99$ by n by $p = 1$ array is available. The index t corresponds to the different points of measurement, the index n to the different observations.

Details

When using this data set please cite both Felipe et al. (2005) and Hubert et al. (2017).

Source

Felipe J.C., Traina A.J.M., Traina C. (2005). Global warp metric distance: boosting content-based image retrieval through histograms. Proceedings of the Seventh IEEE International Symposium on Multimedia (ISM05), p.8.

Chen Y., Keogh E., Hu B., Begum N., Bagnall A., Mueen A., Batista G.J. (2015). The UCR Time Series Classification Archive. [http://www.cs.ucr.edu/~eamonn/time_series_data]

References

Hubert M., Rousseeuw P.J., Segaert P. (2017). Multivariate and functional classification using depth and distance. *Advances in Data Analysis and Classification*, **11**, 445–466.

Examples

```
data(mri)
par(mfrow = c(2,1))
matplot(y = mri$bodypart1[,,1],
        type = "l", col = "black", lty = 1,
        xlab = "", ylab="", main = "bodypart 1")
matplot(y = mri$bodypart2[,,1],
        type = "l", col = "black", lty = 1,
        xlab = "", ylab="", main = "bodypart 2")
par(mfrow = c(1,1))
```

---

octane *Near infrared spectra of gasoline samples*

Description

Near infrared absorbance spectra (NIR) of 39 gasoline samples over 226 wavelengths ranging from 1102nm to 1552nm with measurements every two nanometers. Six of the samples (25, 26, 36-39) contain added alcohol.

Usage

```
data(octane)
```

Format

A three-dimensional $t = 226$ by $n = 39$ by $p = 1$ array.

Source

Esbensen K., Schonkopf S., Midtgaard T. (1994). Multivariate Analysis in Practice. Trondheim: Camo.

References

Hubert M., Rousseeuw P.J., Vanden Branden K. (2005). ROBPCA: a new approach to robust principal component analysis. *Technometrics*, **47**, 64–79.

Examples

```
data(octane)
matplot(octane[,1:30,1], type = "l", lty = 1, col = "black")
```

---

outlyingness *Stahel-Donoho outlyingness of points relative to a dataset*

Description

Computes the Stahel-Donoho outlyingness (SDO) of p-dimensional points z relative to a p-dimensional dataset x. For each multivariate point z_i, its outlyingness relative to x is defined as its maximal univariate Stahel-Donoho outlyingness measured over all directions. To obtain the univariate Stahel-Donoho outlyingness in the direction v, the dataset x is projected on v, and the robustly standardized distance of $v'z_i$ to the robust center of the projected data points xv is computed.

Usage

```
outlyingness(x, z = NULL, options = list())
```

Arguments

x	An n by p data matrix.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the outlyingness. If z is not specified, it is set equal to x.
options	A list of available options:

- type
  Determines the desired type of invariance and should be one of "Affine", "Rotation" or "Shift". When the option "Affine" is used, the directions v are orthogonal to hyperplanes spanned by p observations from x. When the option "Rotation" is used, the directions pass by two randomly selected observations from x. With the option "Shift", directions are randomly generated.
  Defaults to "Affine".
- ndir
  Determines the number of directions v by setting ndir to a specific number or to "all". In the latter case, an exhaustive search over all possible directions (according to type) is performed. If ndir is larger than the number of possible directions, the algorithm will automatically use this setting.
  Defaults to $250p$ when type="Affine", to 5000 when type="Rotation" and to 12500 when type="Shift".

- stand
  Determines how to robustly standardize the projected data: "MedMad" uses the median and the MAD, "unimcd" uses the univariate MCD of location and scale.
  Defaults to "MedMad".
- centered
  When the data matrix x is already centered, no robust center should be computed in each direction. In that case, centered should be set to TRUE.
  Defaults to FALSE.
- h
  When the input argument stand is equal to unimcd, the parameter h controls the number of data points that define the MCD estimator (see covMcd in the robustbase package). This value should lie between $[n/2] + 1$ and n.
  Defaults to $[n/2] + 1$.
- seed
  A strictly positive integer specifying the seed to be used by the C++ code.
  Defaults to 10.

Details

The Stahel-Donoho outlyingness has been introduced by Stahel (1981) and Donoho (1982). It is mostly suited to measure the degree of outlyingness of multivariate points with respect to a data cloud from an elliptical distribution.

Depending on the dimension p, different approximate algorithms are implemented. The affine invariant algorithm can only be used when $n > p$. It draws ndir times at random p observations from x and considers the direction orthogonal to the hyperplane spanned by these p observations. At most p out of n directions can be considered. The orthogonal invariant version can be applied to high-dimensional data. It draws ndir times at random 2 observations from x and considers the direction through these observations. Here, at most 2 out of n directions can be considered. Finally, the shift invariant version randomly draws ndir vectors from the unit sphere.

The resulting Stahel-Donoho outlyingness values are invariant to affine transformations, rotations and shifts respectively provided that the seed is kept fixed at different runs of the algorithm. Note that the SDO values are guaranteed to increase when more directions are considered provided the seed is kept fixed, as this ensures that the random directions are generated in a fixed order.

An observation from x and z is flagged as an outlier if its SDO exceeds a cutoff value. This cutoff value is determined using the procedure in Rousseeuw et al. (2018). First, the logarithm of the SDO values is taken to render their distribution more symmetric, after which a normal approximation yields a cutoff on these values. The cutoff is then transformed back by applying the exponential function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it. Moreover, from the definition of the Stahel-Donoho outlyingness it follows that the outlyingness is ill-defined when the robust scale of the data projected on the direction v equals zero. In this case the algorithm will stop and give a warning. The returned values then include the direction v as well as an indicator specifying which of the observations of x belong to the hyperplane orthogonal to v.

Value

A list with components:

`outlyingnessX`	Vector of length n giving the outlyingness of the observations in `x`.
`outlyingnessZ`	Vector of length m giving the outlyingness of the points in `z`.
`cutoff`	Points whose outlyingness exceeds this cutoff can be considered as outliers with respect to `x`.
`flagX`	Observations of `x` whose outlyingness exceeds the cutoff value receive a flag `FALSE`, regular observations receive a flag `TRUE`.
`flagZ`	Points of `z` whose outlyingness exceeds the cutoff value receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.
`singularSubsets`	When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p - 1$. In such a case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position. When the input parameter type is equal to `"Rotation"` it is possible that two randomly selected points of the data coincide due to ties in the data. In such a case this value signals how many times this happens.
`dimension`	When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.
`hyperplane`	When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust scale of xv is equal to zero, this equals v.
`inSubspace`	When a direction v is found such that the robust scale of xv is zero, the observations from `x` which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

P. Segaert using `C++` code by K. Vakili and P. Segaert.

References

Stahel W.A. (1981). Robuste Schatzungen: infinitesimale Optimalitat und Schatzungen von Kovarianzmatrizen. PhD Thesis, ETH Zurich.

Donoho D.L. (1982). Breakdown properties of multivariate location estimators. Ph.D. Qualifying paper, Dept. Statistics, Harvard University, Boston.

Maronna R.A., Yohai V. (1995). The behavior of the Stahel-Donoho robust multivariate estimator. *Journal of the American Statistical Association*, **90**, 330–341.

Rousseeuw P.J., Raymaekers J., Hubert M., (2018). A measure of directional outlyingness with applications to image data and video. *Journal of Computational and Graphical Statistics*, **27**, 345–359.

See Also

projdepth, projmedian, adjOutl, dirOutl

Examples

```
# Compute the outlyingness of a simple two-dimensional dataset.
# Outliers are plotted in red.

if (requireNamespace("robustbase", quietly = TRUE)) {
    BivData <- log(robustbase::Animals2)
} else {
  BivData <- matrix(rnorm(120), ncol = 2)
  BivData <- rbind(BivData, matrix(c(6,6,6,-2), ncol = 2))
}

Result <- outlyingness(x = BivData)
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")

# The number of directions may be specified through
# the option list. The resulting outlyingness is
# monotone increasing in the number of directions.
Result1 <- outlyingness(x = BivData,
                        options = list(ndir = 50)
                        )
Result2 <- outlyingness(x = BivData,
                        options = list(ndir = 100)
                        )
which(Result2$outlyingnessX - Result1$outlyingnessX < 0)
# This is however not the case when the seed is changed
Result1 <- outlyingness(x = BivData,
                        options = list(ndir = 50)
                        )
Result2 <- outlyingness(x = BivData,
                        options = list(ndir = 100,
                                       seed = 950)
                        )
plot(Result2$outlyingnessX - Result1$outlyingnessX,
     xlab = "Index", ylab = "Difference in outlyingness")

# We can also consider directions through two data
# points. If the sample is small enough one may opt
# to search over all choose(n,2) directions.
# Note that the computational load increases considerably
# as n becomes larger.
Result <- outlyingness(x = BivData,
                       options = list(type = "Rotation",
                                      ndir = "all")
                       )
IndOutliers <- which(!Result$flagX)
plot(BivData)
```

```
points(BivData[IndOutliers,], col = "red")

# Alternatively one may consider randomly generated directions.
Result <- outlyingness(x = BivData,
                       options = list(type = "Shift",
                                      ndir = 1000)
                      )
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")

# The default option of using the MAD for the scale may be
# changed to using the univariate mcd.
Result <- outlyingness(x = BivData,
                       options = list(type = "Affine",
                                      ndir = 1000,
                                      stand = "unimcd",
                                      h = 0.75*nrow(BivData))
                      )
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")
```

`plane` *Fighter plane dataset*

Description

The fighter plane dataset of Thakoor and Gao (2005) describes 7 shapes: of the Mirage, Eurofighter, F-14 with wings closed, F-14 with wings opened, Harrier, F-22 and F-15. Each class contains 30 shape samples obtained from digital pictures, which Thakoor and Gao (2005) then reduced to univariate functions.

Usage

```
data(plane)
```

Format

A list of arrays corresponding to each plane. For each plane, a three-dimensional t by n by $p = 1$ array is available. The index t corresponds to the different measurement points, the index n to the different observations.

Details

When using this data set please cite both Thakoor et al. (2005) and Hubert et al. (2017).

Source

Thakoor N., Gao J. (2005). Shape classifier based on generalized probabilistic descent method with hidden Markov descriptor. Tenth IEEE International Conference on Computer Vision (ICCV 2005), Vol. 1: 495–502.

Chen Y., Keogh E., Hu B., Begum N., Bagnall A., Mueen A., Batista G.J. (2015). The UCR time series classification archive. [http://www.cs.ucr.edu/~eamonn/time_series_data]

References

Hubert M., Rousseeuw P.J., Segaert P. (2017). Multivariate and functional classification using depth and distance. *Advances in Data Analysis and Classification*, **11**, 445–466.

Examples

```
data(plane)
par(mfrow = c(2,1))
matplot(y = plane$plane1[,,1],
        type ="l", col = "black", lty = 1,
        xlab = "", ylab = "", main = "plane 1")
matplot(y = plane$plane2[,,1],
        type = "l", col = "black", lty = 1,
        xlab = "", ylab = "", main = "plane 2")
par(mfrow = c(1,1))
```

---

plotContours *Draws depth contours of bivariate data*

Description

Draws the depth contours of bivariate data computed with `depthContour`.

Usage

```
plotContours(x, depthContour, data = TRUE)
```

Arguments

`x`	An n by 2 data matrix.
`depthContour`	The result of a call to `depthContour`.
`data`	Logical value indicating whether the data `x` should be plotted.

Details

The plot is made using `ggplot2`. The plot itself is returned by the function and is fully customisable using standard `ggplot2` commands.

Author(s)

P. Segaert

References

Ruts I., Rousseeuw P.J. (1996). Computing depth contours of bivariate point clouds. *Computational Statistics & Data Analysis*, **23**, 153-168.

See Also

depthContour

Examples

```
# Compute and plot some halfspace depth contours of a two-dimensional dataset.
# The returned object is a ggplot2 object that may be edited
# using standard ggplot2 commands.
data(cardata90)
Result <- depthContour(x = cardata90,
                       alpha = c(0.125, 0.25))
plot <- plotContours(x = cardata90, depthContour = Result)
plot + ylab("Engine displacement") + xlab("Weight in pounds")

# One may consider different depth functions such as projection depth
# by changing the input parameter 'type' in the depthcontour function.
Result <- depthContour(x = cardata90,
                       alpha = c(0.35, 0.55),
                       type = "projdepth")
plotContours(x = cardata90, depthContour = Result)
```

---

projdepth *Projection depth of points relative to a dataset*

Description

Computes the projection depth of p-dimensional points z relative to a p-dimensional dataset x.

Usage

```
projdepth(x, z = NULL, options = list())
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the projection depth. If z is not specified, it is set equal to x.
options	A list of options to pass to the underlying outlyingness routine. See outlyingness for the full list of options.

Details

Projection depth is based on the Stahel-Donoho outlyingness (SDO) and is computed as $1/(1 + SDO)$. It is mostly suited to measure the degree of outlyingness of multivariate points with respect to a data cloud from an elliptical distribution.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

See `outlyingness` for more details on the computation of the SDO. To visualize the depth of bivariate data one can apply the `mrainbowplot` function. It plots the data colored according to their depth.

The output values of this function are based on the output of the `outlyingness` function. More details can be found there.

Value

A list with components:

`depthX` Vector of length n giving the projection depth of the observations in `x`.

`depthZ` Vector of length m giving the projection depth of the points in `z`.

`cutoff` Points whose projection depth is smaller than this cutoff can be considered as outliers. Equivalently the outliers are those points whose Stahel-Donoho outlyingness exceed the corresponding cutoff.

`flagX` Observations of `x` whose projection depth is smaller than the cutoff receive a flag `FALSE`, regular observations receive a flag `TRUE`.

`flagZ` Points of `z` whose projection depth is smaller than the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.

`singularSubsets` When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p-1$. In that case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position. When the input parameter type is equal to `"Rotation"` it is possible that two randomly selected points of the data coincide due to ties in the data. In this case this value signals how many times this happens.

`dimension` When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.

`hyperplane` When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust scale of xv is equal to zero, this equals v.

`inSubspace` When a direction v is found such that the robust scale of xv is zero, the observations from `x` which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

P. Segaert

References

Zuo Y. (2003). Projection-based depth functions and associated medians. *The Annals of Statistics*, **31**, 1460–1490.

See Also

outlyingness, projmedian, mrainbowplot, adjOutl, dirOutl

Examples

```
# Compute the projection depth of a simple two-dimensional dataset.
# Outliers are plotted in red.

if (requireNamespace("robustbase", quietly = TRUE)) {
    BivData <- log(robustbase::Animals2)
} else {
  BivData <- matrix(rnorm(120), ncol = 2)
  BivData <- rbind(BivData, matrix(c(6,6, 6, -2), ncol = 2))
}

Result <- projdepth(x = BivData)
IndOutliers <- which(!Result$flagX)
plot(BivData)
points(BivData[IndOutliers,], col = "red")

# A multivariate rainbowplot may be obtained using mrainbowplot.
plot.options = list(legend.title = "PD")
mrainbowplot(x = BivData,
             depths = Result$depthX, plot.options = plot.options)

# Options for the underlying outlyingness routine may be passed
# using the options argument.
Result <- projdepth(x = BivData,
                    options = list(type = "Affine",
                                   ndir = 1000,
                                   stand = "MedMad",
                                   h = nrow(BivData)
                                   )
                   )
```

---

projmedian *Location estimates based on projection depth*

Description

Computes a projection depth based location estimate of a p-dimensional dataset x.

Usage

```
projmedian(x, projection.depths = NULL, options = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
projection.depths	Vector containing the projection depth of the observations in x.
options	A list of options to pass to the projdepth routine. See projdepth for more details.

Details

The algorithm depends on the function projdepth to compute the projection depth of the observations in x. If these depth values have already been computed, they can be passed as an optional argument to save computing time. If not, the projection depth values will be computed and the user can pass a list with options to the projdepth function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with components:

max	The point of x with maximal projection depth. If multiple points have maximum depth, their center of gravity is returned.
Huber	A weighted center of gravity of all observations. The weights are defined by the Huber function with parameter alpha = 1 / (1+sqrt(qchisq(0.95, p))) Observations for which the projection depth is more than alpha receive weight 1, other points receive weight (sqrt(qchisq(0.95, p)) / outlyingness)^2.

Author(s)

P. Segaert

References

Maronna, R.A., Yohai, V.J. (1995). The behavior of the Stahel-Donoho robust multivariate estimator. *Journal of the American Statistical Association*, **90**, 330–341.

See Also

outlyingness, projdepth, adjOutl, dirOutl

Examples

```
# Compute a location estimate of a two-dimensional dataset.
if (requireNamespace("robustbase", quietly = TRUE)) {
  BivData <- log(robustbase::Animals2)
} else {
  BivData <- matrix(rnorm(120), ncol = 2)
  BivData <- rbind(BivData, matrix(c(6, 6, 6, -2), ncol = 2))
}

result <- projmedian(x = BivData)
plot(BivData, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)
points(result$Huber, col = "blue", pch = 3, cex = 1.5)

# Options for the underlying projdepth routine may be passed
# using the options argument.
result <- projmedian(x = BivData,
                     options = list(type = "Rotation",
                                    ndir = 100,
                                    stand = "unimcd",
                                    h = 0.75*nrow(BivData)))

plot(BivData, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)
points(result$Huber, col = "blue", pch = 3, cex = 1.5)

# One may also compute the depth values of the observations in the data
# separately. This avoids having to recompute them when computing the median.
depth.result <- projdepth(x = BivData)
result <- projmedian(x = BivData,
                     projection.depths = depth.result$depthX)
```

`rdepth` *Regression depth of hyperplanes*

Description

Computes the regression depth of several hyperplanes with respect to a multiple regression dataset with p explanatory variables. The computation is exact for $p <= 3$. An approximate algorithm is used for $p > 3$.

Usage

```
rdepth(x, z = NULL, ndir = NULL)
```

Arguments

`x`	An n by $p + 1$ regression dataset. The first p columns are the explanatory variables. The last column corresponds to the response variable.

z — An m by $p+1$ matrix containing rowwise the hyperplanes for which to compute the regression depth. The first column should contain the intercepts.
If z is not specified, it is set equal to x.

ndir — Controls the number of directions when the approximate algorithm is used. Defaults to $250p$.

Details

Regression depth has been introduced in Rousseeuw and Hubert (1999). To compute the regression depth of a hyperplane, different algorithms are used. When $p <= 3$ it can be computed exactly. In higher dimensions an approximate algorithm is used (Rousseeuw and Struyf 1998).

It is first checked whether the data x lie in a subspace of dimension smaller than $p + 1$. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with components:

depthZ — A vector containing the regression depth of the hyperplanes in z.

singularSubsets — A vector of length m.
For each hyperplane, it contains the number of singular subsamples when the approximate algorithm is used. The actual number of directions used in the algorithm therefore equals ndir-singularSubsets.

dimension — When the data x are lying in a lower dimensional subspace, the dimension of this subspace.

hyperplane — When the data x are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert using Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf

References

Rousseeuw P.J., Hubert M. (1999). Regression depth. *Journal of the American Statistical Association*, **94**, 388–402.

Rousseeuw P.J., Struyf A. (1998). Computing location depth and regression depth in higher dimensions. *Statistics and Computing*, **45**, 193–203.

See Also

rdepthmedian, cmltest

Examples

```
# Illustrate the concept of regression depth in the case of simple
# linear regression.

data(stars)

# Compute the least squares fit. Due to outliers
# this fit will be bad and thus should result in a small
# regression depth.
temp <- lsfit(x = stars[,1], y = stars[,2])$coefficients
intercept <- temp[1]
slope <- temp[2]
plot(stars, ylim = c(4,7))
abline(a = intercept, b = slope)
abline(a = -9.2, b = 3.2, col = "red")

# Let us compare the regression depth of the least squares fit
# with the depth of the better fitting red regression line.
z <- rbind(cbind(intercept, slope),
           cbind(-9.2, 3.2))
result <- rdepth(x = stars, z = z)
result$depthZ
text(x = 3.8, y = 5.3, labels = round(result$depthZ[1], digits =2))
text(x = 4.45, y = 4.8, labels = round(result$depthZ[2], digits =2), col = "red")

# Compute the depth of some other regression lines to illustrate the concept.
# Note that the stars data set has 47 observations and 1/47 = 0.0212766.
z <- rbind(cbind(6.2, 0),
           cbind(6.5, 0))
result <- rdepth(x = stars, z = z)
abline(a = 6.2, b = 0, col = "blue")
abline(a = 6.5, b = 0, col = "darkgreen")
text(x = 3.8, y = 6.25, labels = round(result$depthZ[1], digits = 2),
     col = "blue")
text(x = 4, y = 6.55, labels = round(result$depthZ[2], digits = 2),
     col = "darkgreen")
# One point needs to removed to make the blue line a nonfit. The green line is
# clearly a nonfit.
```

`rdepthmedian` *Hyperplane of maximal regression depth*

Description

Computes the deepest regression, i.e. the hyperplane with maximal regression depth given a regression dataset with p explanatory variables. The computation is exact for simple regression, and approximate otherwise.

Usage

```
rdepthmedian(x, maxit = NULL)
```

Arguments

x	An n by $p+1$ regression data matrix. The first p columns are the explanatory variables. The last column corresponds to the response variable.
maxit	The maximum number of iterations. Defaults to 100.

Details

In simple regression the deepest regression fit can be computed exactly by considering all lines through two data points and taking the one with maximal regression depth. If several lines have the same maximal regression depth, their average is taken.

When $p > 1$, the approximate MEDSWEEP algorithm is applied (Van Aelst et al, 2002).

It is first checked whether the data lie in a subspace of dimension smaller than $p+1$. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with components:

deepest	A $(p+1)$ vector containing the estimates of the deepest regression fit. The last p values are the slopes, the first value corresponds to the intercept.
depth	The depth of the deepest regression hyperplane.
niter	The number of performed iterations used in the medsweep algorithm.
dimension	When the data x are lying in a lower dimensional subspace, the dimension of this subspace.
hyperplane	When the data x are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert using Fortran code by S. Van Aelst

References

Van Aelst S., Rousseeuw P.J., Hubert M., Struyf A. (2002). The deepest regression method. *Journal of Multivariate Analysis*, **81**, 138–166.

See Also

rdepth, cmltest

Examples

```
# Illustrate the concept of deepest regression line in the case of simple
# linear regression.
data(stars)
plot(stars, pch=16)
result <- rdepthmedian(x = stars)
abline(result$deepest, col="blue", lwd=2)

x <- matrix(rnorm(3000), ncol = 3) + 10
rdepthmedian(x = x)
```

---

sdepth *Simplicial depth of points relative to a dataset*

Description

Computes the simplicial depth of p-dimensional points z relative to a p-dimensional dataset x. Only dimension $p <= 2$ is supported.

Usage

```
sdepth(x, z = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the simplicial depth. If z is not specified, it is set equal to x.

Details

The simplicial depth has been introduced by Liu (1990). The simplicial depth of a point z_i is defined as the number of simplices with vertices in x that contain z_i. Exact computation of the simplicial depth for bivariate data is performed by means of the algorithm described in Rousseeuw and Ruts (1996). To visualize the depth of bivariate data one can apply the mrainbowplot function. It plots the data with coloring according to their depth.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with components:

depthZ	Vector of length m giving the simplicial depth of the points in z.
dimension	When the data x are lying in a lower dimensional subspace, the dimension of this subspace.
hyperplane	When the data x are lying in a lower dimensional subspace, a direction orthogonal to this subspace.

Author(s)

P. Segaert, based on Fortran code by P.J. Rousseeuw, I. Ruts and A. Struyf.

References

Liu R. (1990). On a notion of data depth based on random simplices. *The Annals of Statistics,* **18**, 405–414.

Rousseeuw P.J., Ruts I. (1996). AS 307: Bivariate location depth. *Applied Statistics,* **45**, 516–526.

See Also

mrainbowplot

Examples

```
data(bloodfat)
Result <- sdepth(x = bloodfat)
mrainbowplot(bloodfat, depth = Result$depthZ)
```

---

sprojdepth *Skewness-adjusted projection depth of points relative to a dataset*

Description

Computes the skewness-adjusted projection depth of p-dimensional points z relative to a p-dimensional dataset x.

Usage

```
sprojdepth(x, z = NULL, options = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
z	An optional m by p matrix containing rowwise the points z_i for which to compute the projection depth. If z is not specified, it is set equal to x.
options	A list of options to pass to the underlying adjOutl routine. See adjOutl for the full list of options.

Details

Skewness-adjusted projection depth is based on the adjusted outlyingness and is computed as $1/(1 + AO)$. As adjusted outlyingness extends the Stahel-Donoho outlyingness towards skewed distributions, the skewness-adjusted projection depth is suited for both elliptical distributions and skewed multivariate data.

It is first checked whether the data is found to lie in a subspace of dimension lower than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

See `adjOutl` for more details on the computation of the AO. To visualize the depth of bivariate data one can apply the `mrainbowplot` function. It plots the data colored according to their depth.

The output values of this function are based on the output of the `adjOutl` function. More details can be found there.

Value

A list with components:

`depthX`	Vector of length n giving the skewness-adjusted projection depth of the observations in `x`.
`depthZ`	Vector of length m giving the skewness-adjusted projection depth of the points in `z`.
`cutoff`	Points whose skew-adjusted projection depth is smaller than this cutoff can be considered as outliers.
`flagX`	Observations of `x` whose adjusted outlyingness exceeds the cutoff receive a flag `FALSE`, regular observations receive a flag `TRUE`.
`flagZ`	Points of `z` whose adjusted outlyingness exceeds the cutoff receive a flag equal to `FALSE`, otherwise they receive a flag `TRUE`.
`singularSubsets`	When the input parameter type is equal to `"Affine"`, the number of p-subsets that span a subspace of dimension smaller than $p-1$. In that case the orthogonal direction can not be uniquely determined. This is an indication that the data are not in general position. When the input parameter type is equal to `"Rotation"` it is possible that two randomly selected points of the data coincide due to ties in the data. In this case this value signals how many times this is the case.
`dimension`	When the data `x` are lying in a lower dimensional subspace, the dimension of this subspace.
`hyperplane`	When the data `x` are lying in a lower dimensional subspace, a direction orthogonal to this subspace. When a direction v is found such that the robust skew-adjusted scale of xv is equal to zero, this equals v.
`inSubspace`	When a direction v is found such that AO(xv) is ill-defined, the observations from `x` which belong to the hyperplane orthogonal to v receive a value TRUE. The other observations receive a value FALSE.

Author(s)

P. Segaert with original code from M. Maechler, G. Brys, K. Vakili

References

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Hubert M, Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection. *Statistical Methods & Applications*, **24**, 177–202.

See Also

adjOutl, sprojmedian, mrainbowplot, dirOutl, outlyingness

Examples

```
# Compute the skewness-adjusted projection depth
# of a two-dimensional dataset.

data(bloodfat)
Result <- sprojdepth(x = bloodfat)

# A multivariate rainbowplot may be obtained using mrainbowplot.
plot.options = list(legend.title = "SPD")
mrainbowplot(x = bloodfat,
             depths = Result$depthX, plot.options = plot.options)

# Options for the underlying outlyingness routine may be passed
# using the options argument.
Result <- sprojdepth(x = bloodfat,
                     options = list(type = "Affine",
                                    ndir = 1000,
                                    seed = 12345
                                    )
                    )
```

---

sprojmedian *Location estimates based on skewness-adjusted projection depth*

Description

Computes a skewness-adjusted projection depth based location estimate of a p-dimensional dataset x.

Usage

```
sprojmedian(x, sprojection.depths = NULL, options = NULL)
```

Arguments

x	An n by p data matrix with observations in the rows and variables in the columns.
sprojection.depths	Vector containing the skewness-adjusted projection depth values of the observations in x.
options	A list of options to pass to the sprojdepth routine. See sprojdepth for more details.

Details

The algorithm depends on the function sprojdepth to compute the skewness-adjusted projection depth values of the observations in x. If these depth have already been computed they can be passed as an optional argument to save computing time. If not, the skewness-adjusted projection depth values will be computed and the user can pass a list with options to the sprojdepth function.

It is first checked whether the data lie in a subspace of dimension smaller than p. If so, a warning is given, as well as the dimension of the subspace and a direction which is orthogonal to it.

Value

A list with component:

max	The point of x with maximal skewness-adjusted projection depth. If multiple points have maximum depth, their center of gravity is returned.

Author(s)

P. Segaert

References

Hubert M., Van der Veeken S. (2008). Outlier detection for skewed data. *Journal of Chemometrics*, **22**, 235–246.

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection. *Statistical Methods & Applications*, **24**, 177–202.

See Also

adjOutl, sprojdepth, dirOutl, outlyingness

Examples

```
# Compute a location estimate of a two-dimensional dataset.
data(bloodfat)

result <- sprojmedian(x = bloodfat)
plot(bloodfat, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)
```

```
# Options for the underlying sprojdepth routine may be passed
# using the options argument.
result <- sprojmedian(x = bloodfat,
                      options = list(type = "Rotation",
                                     ndir = 1000
                                     )
                     )
plot(bloodfat, pch = 16)
points(result$max, col = "red", pch = 18, cex = 1.5)

# One may also compute the depth values of the observations in the data
# separately. This avoids having to recompute them when computing the median.
depth.result <- sprojdepth(x = bloodfat)
result <- sprojmedian(x = bloodfat,
                      sprojection.depths = depth.result$depthX)
```

`stars` *stars data*

Description

Data corresponding to a Hertzsprung-Russel diagram of the star cluster CYG OB1 containing 47 stars in the direction of Cygnus. A typical Hertzsprung-Russel diagram shows the logarithm of the temperature in reverse order from high to low.

The data has been taken from Humphreys (1978) by C. Doom who calibrated them according to Vansina and De Greve (1982).

Usage

```
data(stars)
```

Format

A data frame containing the following variables:

`LogTemp` Logarithm of the effective temperature at the star's surface.

`LogLight` Logarithm of a star's light intensity.

Source

Humphreys R.M. (1978). Studies of luminous stars in nearby galaxies. I. Supergiants and O stars in the milky way. *Astrophysics Journal Supplementary Series*, **38**, 309–350.

References

Vansima F., De Greve J.P. (1982). Close binary systems before and after mass transfer. *Astrophysics and Space Science*, **87**, 377–401.

Rousseeuw P.J., Leroy A. (1987). Robust Regression and Outlier Detection. *New York: Wiley*.

Hand D.J., Daly F., Lunn A., McConway A. (1994). A Handbook of Small Data Sets. *Londen: Chapman and Hall*, dataset 367.

Examples

```
data(stars)
plot(stars, xlim = rev(range(stars[,1])))
```

`symtest` *Test for angular symmetry around a specified center for bivariate data*

Description

A test based on halfspace depth for angular symmetry of the bivariate data set `x` around the point `z`. The test is only valid when `x` contains no duplicates.

Usage

```
symtest(x, z, options=list())
```

Arguments

`x`	An n by 2 data matrix.
`z`	A vector of length 2.
`options`	A list of options to pass to `hdepth`. See `hdepth` for more information.

Details

The following hypothesis test is performed:
H_0: The data come from a continuous distribution P which is angularly symmetric about `z`.
The test statistic being used is the halfspace depth of `z` with respect to `x`. Under the null hypothesis the halfspace depth of `z` equals $1/2$. The distribution of the teststatistic under H_0 is $F_n(k) = P(hdepth(P, z) \leq k)$. The p-value of the test is $F_n(k)$ with $k = hdepth(\mathtt{x}, z)$.

Value

`pval`	The p-value of the hypothesis test.

Author(s)

P. Segaert

References

Rousseeuw P.J, Struyf A. (2002). A depth test for symmetry. In: *Goodness-of-Fit Tests and Model Validity*, Birkhäuser Boston, pages 401–412.

Examples

```
# Perform the test on a simple data example.
data(cardata90)
deepest <- hdepthmedian(cardata90)$median
symtest(x = cardata90[!duplicated(cardata90), ], z = deepest)
plot(cardata90)
points(deepest[1], deepest[2], pch=18, col="red", cex=1.2)
```

---

tablets *Near Infrared Spectroscopy responses for a batch of pills*

Description

The original data set consists of Near Infrared Spectroscopy (NIR) data of a batch of pills with different levels of the active compound. This data set considers 70 observations weighing 90 mg and 20 observations weighing 250 mg and also differ in the amount of active compound.

The data correspond to the form discussed in Hubert et al. (2015), see below.

Usage

```
data(tablets)
```

Format

A three dimensional $t = 404$ by $n = 90$ by $p = 3$ array, with t the number of observed time points, n the number of functional observations and p the number of measurements for every functional observation at every wavelength.

Details

For each wavelength and each curve, a three-dimensional vector of observations is available. The second coordinate corresponds to the original data, the first coordinate corresponds to the mean of the corresponding curve. The last component corresponds to the derivative of the original curve data.

When using this data, please cite both of the references listed below.

Source

Dyrby M., Engelsen S.B., Norgaard L., Bruhn M., Nielsen L. (2002). Chemometric quantitation of the active substance in a pharmaceutical tablet Using near infrared (NIR) transmittance and NIR FT-Raman spectra. *Applied Spectroscopy*, **56**, 579–585.

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Examples

```
data(tablets)
par(mfrow = c(3,1))
matplot(tablets[,,1], type = "l", lty = 1, col = "black")
matplot(tablets[,,2], type = "l", lty = 1, col = "black")
matplot(tablets[,,3], type = "l", lty = 1, col = "black")
par(mfrow = c(1,1))
```

`wine` *Proton Nuclear Magnetic Resonance spectra of 40 different wine samples*

Description

The original data set consists of Proton Nuclear Magnetic Resonance (NMR) spectra of 40 different wine samples in the spectral region from 6.00ppm to 0.50ppm. This data set corresponds to the region between wavelengths 5.62ppm and 5.37ppm only for which $t = 397$ measurements are available for each curve. The data has been analyzed in Hubert et al. (2015), see below.

Usage

```
data(wine)
```

Format

A three dimensional $t = 397$ by $n = 40$ by $p = 1$ array, with t the number of observed time points, n the number of functional observations and p the number of measurements for every functional observation at every wavelength.

Details

When using this data set, please cite both of the references below.

Source

Larsen F., van den Berg F., Engelsen S. (2006). An exploratory chemometric study of NMR spectra of table wines. *Journal of Chemometrics*, **20**, 198–208.

References

Hubert M., Rousseeuw P.J., Segaert P. (2015). Multivariate functional outlier detection (with rejoinder). *Statistical Methods & Applications*, **24**, 177–202.

Examples

```
data(wine)
matplot(wine[,,1], type = "l", lty = 1, col = "black")
```

Index