Package: ssMRCD (via r-universe)

October 23, 2024

Type Package

Title Spatially Smoothed MRCD Estimator

Version 1.1.0

Maintainer Patricia Puchhammer <patricia.puchhammer@tuwien.ac.at>

Description Estimation of the Spatially Smoothed Minimum Regularized Determinant (ssMRCD) estimator and its usage in an ssMRCD-based outlier detection method as described in Puchhammer and Filzmoser (2023) <doi:10.1080/10618600.2023.2277875> and for sparse robust PCA for multi-source data described in Puchhammer, Wilms and Filzmoser (2024) <doi:10.48550/arXiv.2407.16299>. Included are also complementary visualization and parameter tuning tools.

License GPL-3

Encoding UTF-8

LazyData true

Imports stats, grDevices, graphics, robustbase, scales, car, dbscan, plot3D, dplyr, ggplot2, expm, foreach, doParallel, rrcov, DescTools, rootSolve, parallel, Matrix, reshape2

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.0.0)

VignetteBuilder knitr

NeedsCompilation no

Author Patricia Puchhammer [aut, cre, cph], Peter Filzmoser [aut]

Repository CRAN

Date/Publication 2024-08-23 11:10:02 UTC

Contents

align_PC *Align Loadings of Principal Components*

Description

Aligns loadings per neighborhood for better visualization and comparison. Different options are available.

Usage

```
align_PC(PC, N, p, type = "largest", vec = NULL)
```

Arguments

PC	matrix of loadings of size Np x k
N	integer, number of groups/neighborhoods
p	integer, number of variables
type	character indicating how loadings are aligned (see details), options are "largest", "maxvar","nonzero","mean", "scalar", "none".
vec	NULL or vector containing vectors for type "scalar"

Details

For input type possible values are "largest", "maxvar","nonzero","mean","scalar". For option "maxvar" the variable with the highest absolute value in the loading is scaled to be positive (per neighborhood, per loading). For option "nonzero" the variable with largest distance to zero in the entries is scaled to be positive (per neighborhood, per loading). For option "scalar" the variable is scaled in a way, that the scalar product between the loading and the respective part of vec is positive (per neighborhood, per loading). If vec is of size p times k, the same vector is used for all neighborhoods. Option "mean" is option "scalar" with vec being the mean of the loadings per variable across neighborhoods. Option "largest" scales the largest absolute value to be positive per neighborhood and per PC. Option "none" does nothing and returns PC.

Value

Returns a matrix of loadings of size Np times k.

Examples

```
x = matrix(c(1, 0, 0, 0, sqrt(0.5), -sqrt(0.5), 0, 0,
             0, sqrt(1/3), -sqrt(1/3), sqrt(1/3), sqrt(0.5), sqrt(0.5), 0, 0),
           ncol = 2)
align_PC(PC = x, N = 2, p = 4, type = "largest")
align_PC(PC = x, N = 2, p = 4, type = "mean")
```

`biplot.PCAloc` *Biplot for PCAloc*

Description

Biplot for PCAloc

Usage

```
## S3 method for class 'PCAloc'
biplot(x, ...)
```

Arguments

`x` object of class PCAloc.

`...` other input arguments, see details.

Details

Additional parameters that can be given to the function are:

`shape` point shape

`size` point size

`alpha` transparency

`color` either "`variable`" or "`groups`" indication how points should be coloured.

Value

Returns version of biplot for PCAloc object.

Examples

```
# set seed
set.seed(236)

# make data
data = matrix(rnorm(2000), ncol = 4)
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariance matrices
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)
```

```
# sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# plot biplot
biplot(pca, alpha = 0.4, shape = 16, size = 2, color = "variable")
```

---

contamination_random *Contamination Through Swapping*

Description

This function swaps observations completely random in order to introduce contamination in the data. Used in parameter_tuning.

Usage

```
contamination_random(cont, data)
```

Arguments

cont	numeric, amount of contamination in data.
data	data whose observations should be switched.

Value

A matrix with switched observations.

Examples

```
# set seed
set.seed(1)

# get data
data(weatherAUT2021)

# switch 5% of observations
contamination_random(cont = 0.05, data = weatherAUT2021[,1:6])
```

eval_objective *Objective function value for local sparse PCA*

Description

Objective function value for local sparse PCA

Usage

```
eval_objective(PC, eta, gamma, COVS)
```

Arguments

PC	vectorised component to evaluate.
eta	degree of sparsity.
gamma	distribution of sparsity between groupwise ($\gamma = 1$) and entrywise ($\gamma = 0$) sparsity.
COVS	list of covariance matrices used for PCA

Value

Returns value of the objective function for given v.

Examples

```
S1 = matrix(c(1, 0.9, 0.8, 0.5,
              0.9, 1.1, 0.7, 0.4,
              0.8, 0.7, 1.5, 0.2,
              0.5, 0.4, 0.2, 1), ncol = 4)
S2 = t(S1)%*% S1
S2 = S2/2

eval_objective(PC = c(1,0,0,0,sqrt(2),0,0,-sqrt(2)),
               eta = 1, gamma = 0.5,
               COVS = list(S1, S2))
```

explained_var *Explained Variance summarized over Groups*

Description

Explained Variance summarized over Groups

Usage

```
explained_var(COVS, PC, k, type = "scaled", cor = FALSE, gamma = 0.5)
```

Arguments

`COVS` list of covariance matrices

`PC` matrix-like object holding the loadings of length np

`k` which component should be evaluated

`type` character, either `"scaled"` for scaling using the extremes solutions or `"percent"` as percentage of overall variance.

`cor` logical, if `COVS` is a correlation matrix or not

`gamma` scalar between 0 and 1 indicatig distribution of sparsity.

Value

Returns scalar

Examples

```
S1 = matrix(c(1, 0.9, 0.8, 0.5,
              0.9, 1.1, 0.7, 0.4,
              0.8, 0.7, 1.5, 0.2,
              0.5, 0.4, 0.2, 1), ncol = 4)
S2 = t(S1)%*% S1
S2 = S2/2

explained_var(COVS = list(S1, S2),
              PC = c(1,0,0,0,sqrt(2),0,0,-sqrt(2)),
              k = 1,
              cor = FALSE,
              gamma = 0.5)

explained_var(COVS = list(cov2cor(S1), cov2cor(S2)),
              PC = c(1,0,0,0,sqrt(2),0,0,-sqrt(2)),
              k = 1,
              cor = TRUE,
              gamma = 0.5)
```

`geo_weights` *Inverse Geographic Weight Matrix*

Description

Calculates a inverse-distance based weight matrix for the function `ssMRCD` (see details).

Usage

```
geo_weights(coordinates, groups)
```

Arguments

`coordinates`	matrix of coordinates of observations.
`groups`	vector of neighborhood groups.

Details

First, the centers (means of the coordinates given) c_i of each neighborhood is calculated. Then, the Euclidean distance between the centers is calculated and the weight is based on the inverse distance between two neighborhoods,

$$w_{ij} = \frac{1}{dist(c_i, c_j)}.$$

It is scaled according to a weight matrix.

Value

Returns a weighting matrix `W` and the coordinates of the centers per neighborhood `centersN`.

See Also

`rescale_weights`

Examples

```
coordinates = matrix(rnorm(1000), ncol = 2, nrow = 500)
groups = sample(1:5, 500, replace = TRUE)

geo_weights(coordinates, groups)
```

`groups_gridbased` *Creates Grid-Based Neighborhood Structure*

Description

This function creates a grid-based neighborhood structure for the `ssMRCD` function using cut-off values for two coordinate axis.

Usage

```
groups_gridbased(x, y, cutx, cuty)
```

Arguments

`x`	vector of first coordinate of data set.
`y`	vector of second coordinate of data set.
`cutx`	cut-offs for first coordinate.
`cuty`	cut-offs for second coordinate.

Value

Returns a neighborhood assignment vector for the coordinates `x` and `y`.

Examples

```
# get data
data(weatherAUT2021)

# set cut-off values
cut_lon = c(9:16, 18)
cut_lat = c(46, 47, 47.5, 48, 49)

# create neighborhood assignments
groups_gridbased(weatherAUT2021$lon,
                      weatherAUT2021$lat,
                      cut_lon,
                      cut_lat)
```

`local_outliers_ssMRCD` *Local Outlier Detection Technique based on ssMRCD*

Description

This function applies the local outlier detection method based on the spatially smoothed MRCD estimator developed in Puchhammer and Filzmoser (2023).

Usage

```
local_outliers_ssMRCD(
  data,
  coords,
  groups,
  lambda,
  weights = NULL,
  k = NULL,
  dist = NULL
)
```

Arguments

`data` data matrix with measured values.

`coords` matrix of coordinates of observations.

`groups` vector of neighborhood assignments.

`lambda` scalar used for spatial smoothing (see also `ssMRCD`).

`weights` weight matrix used in `ssMRCD`.

`k` integer, if given the `k` nearest neighbors per observations are used to calculate next distances. Default value is `k = NULL`.

`dist` scalar, if given the neighbors closer than given distance are used for next distances. If `dist` is given, `dist` is used, otherwise `k` is used.

Value

Returns an object of class `"locOuts"` with following components:

`outliers` indices of found outliers.

`next_distance` vector of next distances for all observations.

`cutoff` upper fence of adjusted boxplot (see `adjbox`) used as cutoff value for next distances.

`coords` matrix of observation coordinates.

`data` matrix of observation values.

`groups` vector of neighborhood assignments.

`k, dist` specifications regarding neighbor comparisons.

`centersN` coordinates of centers of neighborhoods.

`matneighbor` matrix storing information which observations where used to calculate next distance for each observation (

`ssMRCD` object of class `"ssMRCD"` and output of `ssMRCD` covariance estimation.

References

Puchhammer P. and Filzmoser P. (2023): Spatially smoothed robust covariance estimation for local outlier detection. doi:10.48550/arXiv.2305.05371

See Also

See also functions `ssMRCD`, `plot.locOuts`, `summary.locOuts`.

Examples

```
# data construction
data = matrix(rnorm(2000), ncol = 4)
coords = matrix(rnorm(1000), ncol = 2)
groups = sample(1:10, 500, replace = TRUE)
lambda = 0.3

# apply function
outs = local_outliers_ssMRCD(data = data,
                             coords = coords,
                             groups = groups,
                             lambda = lambda,
                              k = 10)
outs
```

objective_matrix *Calculation of Objective Function*

Description

Calculation of the value of the objective function for the ssMRCD for a given list of matrices, lambda and a weighting matrix according to formula (3) in Puchhammer and Filzmoser (2023).

Usage

```
objective_matrix(matrix_list, lambda, weights)
```

Arguments

matrix_list	a list of matrices K_i
lambda	scalar smoothing parameter
weights	matrix of weights

Value

Returns the value of the objective function using matrices K_i.

References

Puchhammer P. and Filzmoser P. (2023): Spatially smoothed robust covariance estimation for local outlier detection. doi:10.48550/arXiv.2305.05371

Examples

```
# construct matrices
k1 = matrix(c(1,2,3,4), nrow = 2)
k2 = matrix(c(1,3,5,7), nrow = 2)

# construct weighting matrix
W = matrix(c(0, 1, 1, 0), nrow = 2)

objective_matrix(list(k1, k2), 0.5, W)
```

parameter_tuning *Optimal Smoothing Parameter for ssMRCD based on Local Outliers*

Description

This function provides insight into the effects of different parameter settings.

Usage

```
parameter_tuning(
  data,
  coords,
  groups,
  lambda = c(0, 0.25, 0.5, 0.75, 0.9),
  weights = NULL,
  k = NULL,
  dist = NULL,
  cont = 0.05,
  repetitions = 5
)
```

Arguments

data	matrix with observations.
coords	matrix of coordinates of these observations.
groups	numeric vector, the neighborhood structure that should be used for ssMRCD.
lambda	scalar, the smoothing parameter.
weights	weighting matrix used in ssMRCD.
k	vector of possible k-values to evaluate.
dist	vector of possible dist-values to evaluate.
cont	level of contamination, between 0 and 1.
repetitions	number of repetitions wanted to have a good picture of the best parameter combination.

Value

Returns a matrix of average false-negative rate (FNR) values and the total number of outliers found by the method as aprox for the false-positive rate. Be aware that the FNR does not take into account that there are also natural outliers included in the data set that might or might not be found. Also a plot is returned representing these average. The best parameter selection depends on the goal of the analysis.

Examples

```
# get data set
data("weatherAUT2021")

# make neighborhood assignments
cut_lon = c(9:16, 18)
cut_lat = c(46, 47, 47.5, 48, 49)
N = ssMRCD::groups_gridbased(weatherAUT2021$lon, weatherAUT2021$lat, cut_lon, cut_lat)
table(N)
N[N == 2] = 1
N[N == 3] = 4
N[N == 5] = 4
N[N == 6] = 7
N[N == 11] = 15
N = as.numeric(as.factor(N))

# tune parameters
set.seed(123)
parameter_tuning(data = weatherAUT2021[, 1:6 ],
                 coords = weatherAUT2021[, c("lon", "lat")],
                 groups = N,
                 lambda = c(0.5, 0.75),
                 k = c(10),
                 repetitions = 1)
```

plot.locOuts *Diagnostic Plots for Local Outlier Detection*

Description

This function plots different diagnostic plots for local outlier detection. It can be applied to an object of class "locOuts" which is the output of the function local_outliers_ssMRCD.

Usage

```
## S3 method for class 'locOuts'
plot(
  x,
  type = c("hist", "spatial", "lines", "3D"),
  colour = "all",
  focus = NULL,
  pos = NULL,
  alpha = 0.3,
  data = NULL,
  add_map = TRUE,
  ...
)
```

Arguments

x — a locOuts object obtained by the function local_outliers_ssMRCD.

type — vector containing the types of plots that should be plotted, possible values c("hist", "spatial", "lines", "3D").

colour — character specifying the color scheme (see details). Possible values "all", "onlyOuts", "outScore".

focus — an integer being the index of the observation whose neighborhood should be analysed more closely.

pos — integer specifying the position of the text "cut-off" in the histogram (see par).

alpha — scalar specifying the transparency level of the points plotted for plot type "spatial", "3D" and "lines".

data — optional data frame or matrix used for plot of type "line". Will be used to plot lines based scaled data instead of the data used for local outlier detection.

add_map — TRUE if a map should be plotted along the line plot (type = "lines").

... — further parameters passed on to base-R plotting functions.

Details

Regarding the parameter type the value "hist" corresponds to a plot of the histogram of the next distances together with the used cutoff-value. When using "spatial" the coordinates of each observation are plotted and colorized according to the color setting. The "lines" plot is used with the index focus of one observation whose out/inlyingness to its neighborhood should by plotted. The whole data set is scaled to the range [0,1] and the scaled value of the selected observation and its neighbors are plotted. Outliers are plotted in orange. The "3D" setting leads to a 3D-plot using the colour setting as height. The view can be adapted using the parameters theta and phi.

For the colour setting possible values are "all" (all next distances are used and colored in an orange palette), "onlyOuts" (only outliers are plotted in orange, inliers are plotted in grey) and "outScore" (the next distance divided by the cutoff value is used to colourize the points; inliers are colorized in blue, outliers in orange).

Value

Returns plots regarding next distances and spatial context.

See Also

local_outliers_ssMRCD

Examples

```
# set seed
set.seed(1)

# make locOuts object
data = matrix(rnorm(2000), ncol = 4)
coords = matrix(rnorm(1000), ncol = 2)
```

```
groups = sample(1:10, 500, replace = TRUE)
lambda = 0.3

# local outlier detection
outs = local_outliers_ssMRCD(data = data,
                             coords = coords,
                             groups = groups,
                             lambda = lambda,
                             k = 10)

# plot results
plot(outs, type = "hist")
plot(outs, type = "spatial", colour = "outScore")
plot(outs, type = "3D", colour = "outScore", theta = 0)
plot(outs, type ="lines", focus = outs$outliers[1])
```

plot.PCAloc *Plotting method PCAloc object*

Description

Plotting method PCAloc object

Usage

```
## S3 method for class 'PCAloc'
plot(
  x,
  type = c("loadings", "screeplot", "scores", "score_distances", "biplot"),
  ...
)
```

Arguments

x	object of class PCAloc
type	character indicating the type of plot, see details.
...	further arguments passed down.

Value

Returns plots in ggplot2.

Examples

```
# set seed
set.seed(236)

# create data and setup
data = matrix(rnorm(2000), ncol = 4)
```

```
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariances
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)

# calculate sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# align loadings
pca$PC = align_PC(PC = pca$PC, N = pca$N, p = pca$p, type = "mean")

# plot different PCA plots
plot(x = pca, type = "score_distances", groups = groups, X = data, ssMRCD = covs, k = 2)
plot(x = pca, type = "biplot", color = "variable")
plot(x = pca, type = "scores", groups = groups, X = data, ssMRCD = covs, k = 1)
plot(x = pca, type = "screeplot")
plot(x = pca, type = "loadings", k = 1)
```

plot.ssMRCD *Plot Method for ssMRCD Object*

Description

Plots diagnostics for function output of ssMRCD regarding convergence behavior and the resulting covariances matrices.

Usage

```
## S3 method for class 'ssMRCD'
plot(
  x,
  type = c("convergence", "ellipses"),
  centersN = NULL,
  colour_scheme = "none",
  xlim_upper = 9,
  manual_rescale = 1,
  legend = TRUE,
  xlim = NULL,
  ylim = NULL,
  ...
)
```

Arguments

x	object of class "ssMRCD".
type	type of plot, possible values are "convergence" and "ellipses". See details.

centersN for plot type "ellipses" a matrix specifying the positions of the centers of the covariance estimation centers, see also geo_weights.

colour_scheme coloring scheme used for plot type "ellipses", either "trace" or "regularity" or "none".

xlim_upper numeric giving the upper x limit for plot type "convergence".

manual_rescale for plot type "ellipses" numeric used to re-scale ellipse sizes.

legend logical, if color legend should be included.

xlim vector of xlim (see par).

ylim vector of ylim (see par).

... further plotting parameters.

Details

For type = "convergence" a plot is produced displaying the convergence behaviour. Each line represents a different initial value used for the c-step iteration. On the x-axis the iteration step is plotted with the corresponding value of the objective function. Not monotonically lines are plotted in red.

For type = "ellipses" and more than a 2-dimensional data setting plotting the exact tolerance ellipse is not possible anymore. Instead the two eigenvectors with highest eigenvalue from the MCD used on the full data set without neighborhood assignments are taken and used as axis for the tolerance ellipses of the ssMRCD covariance estimators. The tolerance ellipse for the global MCD covariance is plotted in grey in the upper left corner. It is possible to set the colour scheme to "trace" to see the overall amount of variabilty and compare the plotted covariance and the real trace to see how much variance is not plotted. For "regularity" the regularization of each covariance is shown.

Value

Returns plots of the ssMRCD methodology and results.

See Also

ssMRCD, summary.ssMRCD,local_outliers_ssMRCD, plot.locOuts

Examples

```
# set seed
set.seed(1)

# create data set
data = matrix(rnorm(2000), ncol = 4)
coords = matrix(rnorm(1000), ncol = 2)
groups = sample(1:10, 500, replace = TRUE)
lambda = 0.3

# calculate ssMRCD by using the local outlier detection method
outs = local_outliers_ssMRCD(data = data,
                             coords = coords,
                             groups = groups,
                             lambda = lambda,
                             k = 10)

# plot ssMRCD object included in outs
plot(x = outs$ssMRCD,
     centersN = outs$centersN,
     colour_scheme = "trace",
     legend = FALSE)
```

`plot_loadings` *Plots of loadings of PCAloc object*

Description

Plots of loadings of PCAloc object

Usage

```
plot_loadings(object, ...)
```

Arguments

`object`	object of class PCAloc
`...`	other input arguments, see details.

Details

Additional parameters that can be given to the function are:

`text`	logical if values should be added as text.
`size`	point size.
`tolerance`	tolerance for rounding to zero.
`k`	integer, which component scores should be plotted.
`groupnames`	names of groups.
`varnames`	names of variables.
`textrotate`	angle of text rotation, if included.

Value

Returns loading heatmap for component k.

Examples

```
# set seed
set.seed(236)

data = matrix(rnorm(2000), ncol = 4)
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariance matrices
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)

# sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# plot score distances
plot_loadings(object = pca,
            k = 1,
            size = 2)
```

plot_scores *Plots of score distribution*

Description

Plots of score distribution

Usage

```
plot_scores(X, PC, groups, ssMRCD, ...)
```

Arguments

X	data matrix.
PC	loadings from PCA.
groups	vector containing group assignments.
ssMRCD	ssMRCD object.
...	other input arguments, see details.

Details

Additional parameters that can be given to the function are:

shape	point shape
size	point size
alpha	transparency
k	integer, which component scores should be plotted

Value

Returns histograms of scores for component k.

Examples

```
# set seed
set.seed(236)

data = matrix(rnorm(2000), ncol = 4)
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariance matrices
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)

# sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# plot score distances
plot_scores(PC = pca$PC,
            groups = groups,
            X = data,
            ssMRCD = covs,
            k = 1,
            alpha = 0.4,
            shape = 16,
            size = 2)
```

plot_score_distances *Distance-distance plot of scores of PCA*

Description

Distance-distance plot of scores of PCA

Usage

```
plot_score_distances(X, PC, groups, ssMRCD, k, ...)
```

Arguments

X	data matrix.
PC	loadings from PCA.
groups	vector containing group assignments.
ssMRCD	ssMRCD object.
k	integer of how many components should be used.
...	other input arguments, see details.

Details

Additional parameters that can be given to the function are:

shape	point shape
size	point size
alpha	transparency

Value

Returns distance-distance plot of orthogonal and score distance.

Examples

```
# set seed
set.seed(236)

data = matrix(rnorm(2000), ncol = 4)
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariance matrices
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)

# sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# plot score distances
plot_score_distances(PC = pca$PC,
                     groups = groups,
                     X = data,
                     ssMRCD = covs,
                     k = 2,
                     alpha = 0.4,
                     shape = 16,
                     size = 2)
```

rescale_weights *Rescale Weight Matrix*

Description

Given a matrix with values for neighborhood influences the function rescales the matrix in order to get an appropriate weight matrix used for the function ssMRCD.

Usage

```
rescale_weights(W)
```

Arguments

`W` weight matrix with diagonals equal to zero and at least one positive entry per row.

Value

An appropriately scaled weight matrix.

See Also

ssMRCD, local_outliers_ssMRCD, geo_weights

Examples

```
W = matrix(c(0, 1, 2,
             1, 0, 1,
             2, 1, 0), nrow = 3)
rescale_weights(W)
```

residuals.ssMRCD *Extracting Residuals from Local Fit*

Description

Extracting Residuals from Local Fit

Usage

```
## S3 method for class 'ssMRCD'
residuals(object, ...)
```

Arguments

`object` ssMRCD object, see ssMRCD.

`...` see details

Details

Other input variables are:

`remove_outliers` logical (default FALSE). If TRUE, only residuals from not outlying observations are calculated. If FALSE

`X` matrix of new data, if data from the ssMRCD object is used.

`groups` vector of groups for new data, if NULL data from the ssMRCD object is used.

`mean` logical (default FALSE), specifying if mean of trimmed observations is returned or all residuals.

If X and groups are provided, alpha is set to one and all residuals are used. If remove_outliers is TRUE, alpha is set to 1 automatically.

Value

Returns either all residuals or the mean of the residual norms lower than the alpha- Quantile.

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
localCovs = ssMRCD(x, weights = W, lambda = 0.5)

# residuals of model
residuals(localCovs, remove_outliers = TRUE, mean = FALSE)

# residuals of new data
residuals(localCovs,
      X = matrix(rnorm(20), ncol = 2, nrow = 10),
      groups = rep(2, 10),
      mean =TRUE)
```

restructure_as_list *Restructure Data Matrix as List*

Description

This function restructures neighborhood information given by a data matrix containing all information and one neighborhood assignment vector. It returns a list of data matrices used in ssMRCD.

Usage

```
restructure_as_list(data, groups)
```

Arguments

data	data matrix with all observations.
groups	numeric neighborhood assignment vector.

Value

Returns a list containing the observations per neighborhood assignment. The list is sorted according to the order of the first appearance in the groups vector.

Examples

```
# data matrix
data = matrix(rnorm(n = 3000), ncol = 3)
N_assign = sample(x = 1:10, size = 1000, replace = TRUE)

restructure_as_list(data, N_assign)
```

scale_ssMRCD *Scale Data Locally*

Description

Scale Data Locally

Usage

```
scale_ssMRCD(
  ssMRCD,
  X = NULL,
  groups = NULL,
  multivariate = FALSE,
  center_only = FALSE
)
```

Arguments

`ssMRCD` `ssMRCD` object, see `ssMRCD`

`X` matrix, new data to scale with ssMRCD estimation.

`groups` vector, group assignments of new data `X`.

`multivariate` logical, `TRUE` if multivariate structure should be used. Otherwise, univariate variances from the ssMRCD estimator is used.

`center_only` logical, if `TRUE` observations are only centered.

Value

Returns matrix of observations. If `X = NULL` X from the ssMRCD object is used and sorted according to group numbering.

See Also

`ssMRCD`

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
localCovs = ssMRCD(x, weights = W, lambda = 0.5)

# scale used data
scale_ssMRCD(localCovs,
      multivariate = TRUE)

# scale new data
scale_ssMRCD(localCovs,
      X = matrix(rnorm(20), ncol = 2, nrow = 10),
      groups = rep(2, 10),
      multivariate =TRUE)
```

`scores` *Calculate Scores for local sparse PCA*

Description

Calculate Scores for local sparse PCA

Usage

```
scores(X, PC, groups, ssMRCD = NULL)
```

Arguments

`X`	data set as matrix.
`PC`	loading matrix.
`groups`	vector of grouping structure (numeric).
`ssMRCD`	ssMRCD object used for scaling `X`. If `NULL` no scaling and centering is performed.

Value

Returns a list with scores and univariately and locally centered and scaled observations.

See Also

`ssMRCD`, `scale_ssMRCD`

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
loccovs = ssMRCD(x, weights = W, lambda = 0.5)

# calculate PCA
pca = sparsePCAloc(eta = 1, gamma = 0.5, cor = FALSE,
                   COVS = loccovs$MRCDcov,
                   increase_rho = list(FALSE, 20, 1))

# calculate scores
scores(X = rbind(x1, x2), PC = pca$PC,
       groups = rep(c(1,2), each = 100), ssMRCD = loccovs)
```

`scores.OD` *Orthogonal Distances for PCAloc*

Description

Orthogonal Distances for PCAloc

Usage

```
scores.OD(X, PC, groups, ssMRCD)
```

Arguments

`X`	data matrix of observations.
`PC`	loadings of sparse local PCA.
`groups`	grouping vector for locality.
`ssMRCD`	ssMRCD object used for PCA calculation.

Value

Returns vector of orthogonal distances of observations.

See Also

`scores`, `scores.SD`, `sparsePCAloc`, `scale_ssMRCD`

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
loccovs = ssMRCD(x, weights = W, lambda = 0.5)

# calculate PCA
pca = sparsePCAloc(eta = 1, gamma = 0.5, cor = FALSE,
                   COVS = loccovs$MRCDcov,
                   increase_rho = list(FALSE, 20, 1))

# calculate scores
scores.OD(X = rbind(x1, x2), PC = pca$PC,
          groups = rep(c(1,2), each = 100), ssMRCD = loccovs)
```

`scores.SD` *Score Distances for PCAloc*

Description

Score Distances for PCAloc

Usage

```
scores.SD(X, PC, groups, ssMRCD)
```

Arguments

X	data matrix of observations.
PC	loadings of sparse local PCA.
groups	grouping vector for locality.
ssMRCD	ssMRCD object used for PCA calculation.

Value

Returns vector of score distances of observations.

See Also

scores, scores.OD, sparsePCAloc, scale_ssMRCD

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
loccovs = ssMRCD(x, weights = W, lambda = 0.5)

# calculate PCA
pca = sparsePCAloc(eta = 1, gamma = 0.5, cor = FALSE,
                   COVS = loccovs$MRCDcov,
                   increase_rho = list(FALSE, 20, 1))

# calculate scores
scores.SD(X = rbind(x1, x2), PC = pca$PC,
          groups = rep(c(1,2), each = 100), ssMRCD = loccovs)
```

screeplot.PCAloc *Screeplot for PCAloc*

Description

Screeplot for PCAloc

Usage

```
## S3 method for class 'PCAloc'
screeplot(x, ...)
```

Arguments

`x`	object of class PCAloc.
`...`	other input arguments, see details.

Details

Additional parameters that can be given to the function are:

`text`	logical if text should be plotted
`size`	text size
`cutoff`	cutoff line for scree plot
`groupnames`	name of groups
`textrotate`	angle of text, if text is plotted.

Value

Returns version of scree plot and cumulative explained variance per group for PCAloc object.

Examples

```
# set seed
set.seed(236)
data = matrix(rnorm(2000), ncol = 4)
groups = sample(1:10, 500, replace = TRUE)
W = time_weights(N = 10, c(3,2,1))

# calculate covariance matrices
covs = ssMRCD(data, groups = groups, weights = W, lambda = 0.3)

# sparse PCA
pca = sparsePCAloc(eta = 0.3, gamma = 0.7, cor = FALSE, COVS = covs$MRCDcov,
             n_max = 1000, increase_rho = list(TRUE, 50, 1), trace = FALSE)

# plot biplot
screeplot(pca, text = TRUE, cutoff = 0.8, size = 2)
```

select_smoothing *Optimal Smoothing Parameter for ssMRCD based on Residuals*

Description

The optimal smoothing value for the ssMRCD estimator is based on the residuals and the trimmed mean of the norm.

Usage

```
select_smoothing(
  X,
  groups,
  weights,
  lambda = seq(0, 1, 0.1),
  TM = NULL,
  alpha = 0.75,
  seed = 123436,
  return_all = TRUE,
  cores = 1
)
```

Arguments

X	data matrix containing observations.
groups	grouping vector corresponding to X.
weights	weight matrix for groups, see rescale_weights, and geo_weights.
lambda	vector of parameter values for smoothing, between 0 and 1.
TM	target matrix, if not given MCD (or MRCD if non regular) is used with default values and alpha.
alpha	percentage of outliers to be expected.
seed	seed for ssMRCD calculations.
return_all	logical, if FALSE the function returns only the optimal lambda.
cores	integer, number of cores used for parallel computing.

Value

lambda_opt	optimal lambda for smoothing.
COVS	ssMRCD object with optimal parameter setting.
plot	plot for optimal parameter setting.
residuals	mean of norm of residuals for varying lambda.

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

select_smoothing (X = rbind(x1, x2),
                 groups = rep(c(1,2), each = 100),
                 weights = W,
                 lambda = seq(0, 1, 0.1),
                 return_all = TRUE,
                 cores = 1)
```

select_sparsity *Optimal Sparsity Parameter Selection for PCA*

Description

Optimal Sparsity Parameter Selection for PCA

Usage

```
select_sparsity(
  COVS,
  k = 1,
  rho = NULL,
  cor = FALSE,
  eta = seq(0, 5, by = 0.2),
  gamma = seq(0, 1, 0.05),
  eps_threshold = 0.001,
  eps_root = 0.1,
  eps_ADMM = 1e-04,
  n_max = 300,
  adjust_eta = FALSE,
  cores = 1,
  increase_rho = list(TRUE, 100, 1),
  convergence_plot = FALSE,
  trace = FALSE,
  stop.sparse = TRUE
)
```

Arguments

`COVS` list of covariance or correlation matrices.

`k` number of components to be returned.

`rho` penalty parameter for ADMM.

`cor` logical, if starting values for covariances or correlation matrices should be used.

`eta` vector of possible values for degree of sparsity.

`gamma` vector of possible values for distribution of sparsity. If only one value is provided, the optimal eta is calculated.

`eps_threshold` tolerance for thresholding.

`eps_root` tolerance for root finder.

`eps_ADMM` tolerance for ADMM iterations.

`n_max` maximal number of ADMM iterations.

`adjust_eta` if eta should be adjusted for further components.

`cores` number of cores for parallel computing.

`increase_rho` list of settings for improved automated calculation and convergence. See Details.

`convergence_plot` logical, if convergence plot should be plotted. Not applicable for `cores > 1`.

`trace` logical, if messages should be displayed. Not applicable for `cores > 1`.

`stop.sparse` calculate if AUC should be calculated for PCAs until full sparsity is reached (`TRUE`) or over the whole eta range (`FALSE`). Set to `TRUE`.

Details

The input `increase_rho` consists of a logical indicating if rho should be adjusted if algorithm did not converged within the given maximal number of iterations. Two integers specify the maximal `rho` that is allowed and the step size.

Value

Returns list with

`PCA` object of type PCAloc.

`PC` local loadings of PCA

`gamma` optimal value for gamma.

`eta` optimal value for eta.

`eta_tpo` values of Trade-Off-Product for eta from optimization process.

`auc` area under the curve for varying gamma values.

`pars` parameters and respective sparsity entrywise and mixed and explained variance.

`plot` ggplot object for optimal parameter selection.

`plot_info` additional data for plotting functions.

Examples

```
C1 = matrix(c(1,0,0,0.9), ncol = 2)
C2 = matrix(c(1.1, 0.1, 0.1, 1), ncol = 2)
C3 = matrix(c(1.2, 0.2, 0.2, 1), ncol = 2)

select_sparsity(COVS = list(C1, C2, C3),
                k = 1,
                rho = 5,
                eta = c(0, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5 ,0.75,  1),
                gamma =c(0, 0.25, 0.5, 0.75, 1),
                eps_threshold = 0.005,
                increase_rho = list(FALSE, 20, 5))
```

sparsePCAloc *Calculate Sparse Principle Components*

Description

Calculate Sparse Principle Components

Usage

```
sparsePCAloc(
  eta,
  gamma,
  COVS,
  cor = FALSE,
  rho = NULL,
  k = NULL,
  eps_threshold = NULL,
  eps_ADMM = 1e-04,
  n_max = 200,
  eps_root = 0.1,
  maxiter_root = 50,
  increase_rho = list(TRUE, 100, 1),
  convergence_plot = TRUE,
  starting_value = NULL,
  adjust_eta = TRUE,
  trace = TRUE
)
```

Arguments

eta	numeric, degree of sparsity.
gamma	numeric, distribution of sparsity.
COVS	list of covariance or correlation matrices.
cor	logical, if starting value for correlation or covariance matrices should be used.
rho	numeric bigger than zero, penalty for ADMM.
k	number of components to calculate.
eps_threshold	tolerance for thresholding.
eps_ADMM	tolerance for ADMM convergence.
n_max	number of maximal iterations.
eps_root	tolerance for root finder.
maxiter_root	maximal number of iterations for root finder.
increase_rho	list with entries for stable convergence. See Details.
convergence_plot	logical, if convergence plot should be displayed.
starting_value	optional given starting value.
adjust_eta	logical, if eta should be adjusted by the variance.
trace	logical, if messages should be displayed.

Details

The input increase_rho consists of a logical indicating if rho should be adjusted if algorithm did not converged within the given maximal number of iterations. Two integers specify the maximal rho that is allowed and the step size.

Value

An object of class "PCAloc" containing the following elements:

PC	Matrix of dimension Np x k of stacked loading vectors.
p	Number of variables.
N	Number of neighborhoods.
k	Number of components.
COVS	List of covariance matrices sorted by neighborhood.
gamma	Sparsity distribution.
eta	Amount of sparsity.

converged Logical, if ADMM converged with given specifications.

n_steps Number of steps used.

summary Description of result per component.

residuals Primary and secondary residuals.

Examples

```
C1 = diag(c(1.1, 0.9, 0.6))
C2 = matrix(c(1.1, 0.1, -0.1,
              0.1, 1.0, -0.2,
             -0.1, -0.2, 0.7), ncol = 3)
C3 = (C1 + C2)/2

sparsePCAloc(eta = 1, gamma = 0.5, cor = FALSE, COVS = list(C1, C2, C3),
             n_max = 100, increase_rho = list(FALSE, 100, 1))
```

sparsity_entries *Entry-wise Sparsity in the Loadings*

Description

Entry-wise Sparsity in the Loadings

Usage

```
sparsity_entries(PC, N, p, tolerance = 0, k = 1, scaled = TRUE)
```

Arguments

PC matrix-like object of PCs.

N integer, number of groups.

p integer, number of variables.

tolerance tolerance for sparsity.

k integer or integer vector of which component should be used.

scaled logical, if total number or percentage of possible sparse entries should be returned.

Value

Returns either a percentage (scaled = TRUE) or the amount of zero-values entries (scaled = FALSE).

Examples

```
PC = matrix(c(1,0,2,3,0,7,0,1,0,1,0.001,0), ncol = 2)
sparsity_entries(PC, N = 2, p = 3, tolerance = 0, k = 1, scaled = FALSE)
sparsity_entries(PC, N = 2, p = 3, tolerance = 0.001, k = 2, scaled = TRUE)
```

sparsity_group *Group-wise Sparsity in the Loadings*

Description

Group-wise Sparsity in the Loadings

Usage

```
sparsity_group(PC, N, p, tolerance = 0, k = 1, scaled = TRUE)
```

Arguments

PC	matrix-like object of PCs.
N	integer, number of groups.
p	integer, number of variables.
tolerance	tolerance for sparsity.
k	integer, which components should be used. Does not work for multiple PCs simultaneously.
scaled	logical, if total number or percentage of possible sparse entries should be returned.

Value

Returns either a matrix of percentages (scaled = TRUE) or the amounts of zero-values entries (scaled = FALSE) for each group/neighborhood.

Examples

```
PC = matrix(c(1,0,2,3,0,7,0,1,0,1,0.001,0), ncol = 2)
sparsity_group(PC, N = 2, p = 3, tolerance = 0, k = 1, scaled = FALSE)
sparsity_group(PC, N = 2, p = 3, tolerance = 0.001, k = 2, scaled = TRUE)
```

`sparsity_mixed` *Mixed Sparsity of the Loadings*

Description

Mixed Sparsity of the Loadings

Usage

```
sparsity_mixed(PC, p, N, k = 1, tolerance = 0.001, mean = "arithmetic")
```

Arguments

`PC`	matrix-like object of PCs.
`p`	integer, number of variables.
`N`	integer, number of groups.
`k`	integer, which components should be used. Does not work for multiple PCs simultaneously.
`tolerance`	tolerance for sparsity.
`mean`	if `"arithmetic"` or `"geometric"` mean should be used.

Value

Returns the geometric mean of the percentage of entry-wise and group-wise sparsity.

Examples

```
PC = matrix(c(1,0,2,3,0,7,0,1,0,1,0.001,0), ncol = 2)
sparsity_mixed(PC, N = 2, p = 3, tolerance = 0, k = 1)
sparsity_mixed(PC, N = 2, p = 3, tolerance = 0.001, k = 2, mean = "geometric")
```

`sparsity_summary` *Entry-wise Sparsity in the Loadings per Group*

Description

Entry-wise Sparsity in the Loadings per Group

Usage

```
sparsity_summary(PC, N, p, tolerance = 0, k = 1, scaled = FALSE)
```

Arguments

`PC` matrix-like object of PCs.

`N` integer, number of groups.

`p` integer, number of variables.

`tolerance` tolerance for sparsity.

`k` integer or integer vector of which component should be used.

`scaled` logical, if total number or percentage of possible sparse entries should be returned.

Value

Returns either a matrix of percentages (`scaled = TRUE`) or the amounts of zero-values entries (`scaled = FALSE`) for each group/neighborhood.

Examples

```
PC = matrix(c(1,0,2,3,0,7,0,1,0,1,0.001,0), ncol = 2)
sparsity_summary(PC, N = 2, p = 3, tolerance = 0, k = 1, scaled = FALSE)
sparsity_summary(PC, N = 2, p = 3, tolerance = 0.001, k = 2, scaled = TRUE)
```

`ssMRCD` *Spatially Smoothed MRCD Estimator*

Description

The ssMRCD function calculates the spatially smoothed MRCD estimator from Puchhammer and Filzmoser (2023).

Usage

```
ssMRCD(
  x,
  groups = NULL,
  weights,
  lambda,
  TM = NULL,
  alpha = 0.75,
  maxcond = 50,
  maxcsteps = 200,
  n_initialhsets = NULL
)
```

Arguments

x a list of matrices containing the observations per neighborhood sorted which can be obtained by the function restructure_as_list, or matrix or data frame containing data. If matrix or data.frame, group vector has to be given.

groups vector of neighborhood assignments

weights weighting matrix, symmetrical, rows sum up to one and diagonals need to be zero (see also geo_weights or rescale_weights .

lambda numeric between 0 and 1.

TM target matrix (optional), default value is the covMcd from robustbase.

alpha numeric, proportion of values included, between 0.5 and 1.

maxcond optional, maximal condition number used for rho-estimation.

maxcsteps maximal number of c-steps before algorithm stops.

n_initialhsets number of initial h-sets, default is 6 times number of neighborhoods.

Value

An object of class "ssMRCD" containing the following elements:

MRCDcov List of ssMRCD-covariance matrices sorted by neighborhood.

MRCDicov List of inverse ssMRCD-covariance matrices sorted by neighborhood.

MRCDmu List of ssMRCD-mean vectors sorted by neighborhood.

mX List of data matrices sorted by neighborhood.

N Number of neighborhoods.

mT Target matrix.

rho Vector of regularization values sorted by neighborhood.

alpha Scalar what percentage of observations should be used.

h Vector of how many observations are used per neighborhood, sorted.

numiter The number of iterations for the best initial h-set combination.

c_alpha Consistency factor for normality.

weights The weighting matrix.

lambda Smoothing factor.

obj_fun_values A matrix with objective function values for all initial h-set combinations (rows) and iterations (columns).

`best6pack` initial h-set combinations with best objective function value after c-step iterations.

`Kcov` returns MRCD-estimates without smoothing.

References

Puchhammer P. and Filzmoser P. (2023): Spatially smoothed robust covariance estimation for local outlier detection. doi:10.48550/arXiv.2305.05371

See Also

`plot.ssMRCD`, `summary.ssMRCD`, `restructure_as_list`

Examples

```
# create data set
x1 = matrix(runif(200), ncol = 2)
x2 = matrix(rnorm(200), ncol = 2)
x = list(x1, x2)

# create weighting matrix
W = matrix(c(0, 1, 1, 0), ncol = 2)

# calculate ssMRCD
ssMRCD(x, weights = W, lambda = 0.5)
```

`summary.locOuts` *Summary of Local Outlier Detection*

Description

Prints a summary of the locOuts object obtained by the function `local_outliers_ssMRCD`.

Usage

```
## S3 method for class 'locOuts'
summary(object, ...)
```

Arguments

`object` a locOuts object.

`...` further parameters passed on.

Value

Prints a summary of the `locOuts` object.

See Also

`plot.locOuts`

Examples

```
# set seed
set.seed(1)

# make locOuts object
data = matrix(rnorm(2000), ncol = 4)
coords = matrix(rnorm(1000), ncol = 2)
groups = sample(1:10, 500, replace = TRUE)
lambda = 0.3

# local outlier detection
outs = local_outliers_ssMRCD(data = data,
                             coords = coords,
                             groups = groups,
                             lambda = lambda,
                             k = 10)

# summary method
summary(outs)
```

`summary.PCAloc` *Summary method for PCAloc*

Description

Summary method for PCAloc

Usage

```
## S3 method for class 'PCAloc'
summary(object, ...)
```

Arguments

`object`	object of class PCAloc
`...`	other input variables.

Value

Summary for PCAloc

See Also

`sparsePCAloc`

Examples

```
#'
C1 = diag(c(1.1, 0.9, 0.6))
C2 = matrix(c(1.1, 0.1, -0.1,
              0.1, 1.0, -0.2,
             -0.1, -0.2, 0.7), ncol = 3)
C3 = (C1 + C2)/2

pca = sparsePCAloc(eta = 1, gamma = 0.5, cor = FALSE, COVS = list(C1, C2, C3),
             n_max = 100, increase_rho = list(FALSE, 100, 1), trace = FALSE)

summary(pca)
```

summary.ssMRCD *Summary Method for ssMRCD Object*

Description

Summarises most important information of output ssMRCD.

Usage

```
## S3 method for class 'ssMRCD'
summary(object, ...)
```

Arguments

object	object of class "ssMRCD", output of ssMRCD.
...	further parameters.

Value

Prints a summary of the ssMRCD object.

See Also

See also ssMRCD, plot.ssMRCD.

`time_weights` *Band weight matrix for time series groupings*

Description

Band weight matrix for time series groupings

Usage

```
time_weights(N, off_diag)
```

Arguments

`N`	number of groups.
`off_diag`	vector for off-diagonal values unequal to zero.

Value

Returns weight matrix for time series groups appropriate for `ssMRCD`.

See Also

`geo_weights`, `rescale_weights`

Examples

```
time_weights(N = 10, off_diag = c(2,1))
```

`weatherAUT2021` *Austrian Weather Data 2021*

Description

This data is a subset of the GeoSphere Austria monthly weather data of 2021 averaged using the median. Stations with missing values are removed.

Usage

```
weatherAUT2021
```

Format

A data frame with 183 rows and 10 columns:

name Unique name of the weather station in German.

lon, lat Longitude and latitude of the weather station.

alt Altitude of the weather station (meter).

p Average air pressure (hPa).

s Monthly sum of sunshine duration (hours).

vv Wind velocity (meter/second).

t Air temperature in 2 meters above the ground in (°C).

rsum Average daily sum of precipitation (mm).

rel Relative air humidity (percent).

Source

The original data was downloaded here (December 2022): https://data.hub.geosphere.at/dataset/klima-v1-1m.

References

Data Source: GeoSphere Austria - https://data.hub.geosphere.at.

Examples

```
data(weatherAUT2021)
summary(weatherAUT2021)
```

---

weatherHoheWarte *Vienna Weather Time Series (1960-2023)*

Description

This data is a subset of the GeoSphere Austria daily weather data of the time 1960-2023 for the weather station Hohe Warte in Vienna.

Usage

```
weatherHoheWarte
```

Format

A data frame with 23372 rows and 18 columns including 13 weather measurements:

time Time of measurement in date format.

cloud_cover Daily mean of cloud coverage, values between 1 and 100.

global_radiation Daily sum of global radiation (J/cm^2).

vapor_pressure Daily mean of vapour pressuer (hPa).

max_wind_speed Maximal wind speed (m/s).

air_pressure Daily mean of air pressure (hPa).

relative_humidity Daily mean of relative humidity (percent).

precipitation Daily sum of precepitation (mm).

sight Sight distance at 1pm (m).

sunshine_duration Daily sum of sunshine duration (h).

temperature_max Daily maximum of temperature at 2m air height (°C).

temperature_min Daily minimum of temperature at 2m air height (°C).

temperature_mean Daily mean of temperature at 2m air height (°C).

wind_velocity Daily mean of wind speed (m/s).

year Year of measurement.

month Month of measurement.

day Day of the year of measurement.

season Season of measuremen (1 = winter, 2 = spring, 3 = summer, 4 = fall).

Source

The original data was downloaded here (April 2024): `https://data.hub.geosphere.at/dataset/klima-v2-1d`.

References

Data Source: GeoSphere Austria - `https://data.hub.geosphere.at`.

Examples

```
data(weatherHoheWarte)
summary(weatherHoheWarte)
```

Index