Package 'Numero'

September 14, 2020

Type Package

Title Statistical Framework to Define Subgroups in Complex Datasets

Version 1.7.0

Date 2020-09-13

Author Song Gao [aut],
Stefan Mutter [aut],
Aaron E. Casey [aut],
Ville-Petteri Makinen [aut, cre]

Maintainer Ville-Petteri Makinen <vpmakine@gmail.com>

Description High-dimensional datasets that do not exhibit a clear intrinsic clustered structure pose a challenge to conventional clustering algorithms. For this reason, we developed an unsupervised framework that helps scientists to better subgroup their datasets based on visual cues, please see Gao S, Mutter S, Casey A, Makinen V-P (2019) Numero: a statistical framework to define multivariable subgroups in complex population-based datasets, Int J Epidemiology, 48:369-37, <doi:10.1093/ije/dyy113>. The framework includes the necessary functions to construct a self-organizing map of the data, to evaluate the statistical significance of the observed data patterns, and to visualize the results.

License GPL (>= 2)

Imports Rcpp (>= 0.11.4)

LinkingTo Rcpp

VignetteBuilder knitr

Suggests knitr, rmarkdown

NeedsCompilation yes

Repository CRAN

SystemRequirements C++11

Encoding UTF-8

LazyData true

Date/Publication 2020-09-14 05:10:02 UTC

R topics documented:

`nroAggregate` *Regional averages on a self-organizing map*

Description

Estimate district averages based on assigned map locations for each data point.

Usage

```
nroAggregate(topology, districts, data = NULL)
```

Arguments

`topology`	A data frame with K rows and six columns, see details.
`districts`	An integer vector of M best-matching districts.
`data`	A vector of M elements or an M x N matrix of data values.

Details

Topology can be either the output from `nroKohonen()` or a data frame in the same format as the element `topology` within the the output from `nroKohonen()`.

The input argument `districts` is expected to be the output from `nroMatch()`.

Value

If the input argument `data` is empty, the histogram of the data points on the map is returned (a vector of K elements).

If data are available, a matrix of K rows and N columns that contains the average district values after smoothing is returned. The output includes the attribute 'histogram' that contains data point counts over each data column. Column names and the attribute 'binary' are copied from the input.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)

# Assign data points into districts.
matches <- nroMatch(centroids = sm, data = trdata)

# District averages for one variable.
chol <- nroAggregate(topology = sm, districts = matches,
                     data = dataset$CHOL)
print(chol)

# District averages for all variables.
planes <- nroAggregate(topology = sm, districts = matches, data = dataset)
print(head(planes))
```

`nroColorize` *Assign colors based on value*

Description

Assign colors to map districts based on the respective district values.

Usage

```
nroColorize(values, ranges = NULL, amplitudes = 1, palette = "rhodo")
```

Arguments

values	A vector of K values or a K x N data frame, where K is the number of map districts and N is the number of variables.
ranges	A data frame with N rows and 2 columns, see details.
amplitudes	Single value or a vector of N elements or a data frame of N rows that contains the column AMPLITUDE.
palette	One of pre-defined colormap names (see details).

Details

The argument ranges sets the minimum and maximum district values irrespective of the contents of values. It can be used as a fixed reference across different colorings to ensure that the same value produces the same color across function calls.

The argument amplitudes controls the proportion of the color range that is available for the district value range. For proportions below 1, the minimum district value is assigned to a color that is between the first and middle element in the color palette, and the maximum is assigned to a color that is between the middle and the last element. If amplitude is greater than 1, extreme low and high values are clipped to the first and last color in the palette, respectively.

Available color palettes include "grey", "fire", "jungle", "miami", "rhodo" or "tan". Any other word will revert to a rainbow colormap.

Value

A matrix of hexadecimal color codes as strings. The output also includes the attribute 'contrast' that indicates which colors have a good contrast with black as opposed to white, and the attribute 'ranges' that contains a copy of the dynamic ranges across districts.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)

# Assign data points into districts.
```

```
matches <- nroMatch(centroids = sm, data = trdata)

# District averages for all variables.
planes <- nroAggregate(topology = sm, districts = matches, data = dataset)

# District colors for cholesterol.
chol <- nroColorize(values = planes[,"CHOL"])
print(head(chol))

# District colors for all variables.
colrs <- nroColorize(values = planes)
print(head(colrs))
```

`nroDestratify` *Mitigate data stratification*

Description

Removes differences in value distribution within subsets of data points.

Usage

```
nroDestratify(data, labels)
```

Arguments

`data`	A matrix or a data frame with M rows.
`labels`	A vector of M subset labels.

Details

Only non-binary numerical columns are processed, the rest will be excluded from the results.

The de-stratification algorithm is based on ranked data: the distribution of each subset will be mapped to the pooled distribution over all subsets by matching subset-specific ranking with ranking of all values.

Value

A matrix of de-stratified values. The output also includes the attribute 'incomplete' that lists those columns where (some of) the values were set to missing due to processing failures.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Remove sex differences for creatinine.
creat <- nroDestratify(dataset$CREAT, dataset$MALE)
```

```
# Compare creatinine distributions.
men <- which(dataset$MALE == 1)
women <- which(dataset$MALE == 0)
print(summary(dataset[men,"CREAT"]))
print(summary(dataset[women,"CREAT"]))
print(summary(creat[men]))
print(summary(creat[women]))

# Remove sex differences (produces warnings for binary traits).
ds <- nroDestratify(dataset, dataset$MALE)

# Compare HDL2C distributions.
print(summary(dataset[men,"HDL2C"]))
print(summary(dataset[women,"HDL2C"]))
print(summary(ds[men,"HDL2C"]))
print(summary(ds[women,"HDL2C"]))
```

`nroImpute` *Impute missing values*

Description

Find nearest neighbors by Euclidean distance and impute missing values.

Usage

```
nroImpute(data, subsample = 500, standard = TRUE, message = NULL)
```

Arguments

`data`	A matrix or a data frame.
`subsample`	Maximum number of matchings to test per imputed row.
`standard`	If `TRUE`, the scales of variables are standardized for processing.
`message`	If positive, progress information is printed at the specified interval in seconds.

Details

Non-numeric columns are excluded from processing and returned unaltered.

If `subsample` is less than the number of rows, an equivalent number of randomly picked rows is selected to find the nearest neighbor.

Value

A matrix where missing values have been imputed.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Convert identities to strings (produces a warning later).
ds <- dataset
ds[,"INDEX"] <- paste("K", ds[,"INDEX"], sep=".")

# Introduce missing values to cholesterol.
missing <- seq(from = 1, to = nrow(ds), length.out = 40)
missing <- unique(round(missing))
ds[missing,"CHOL"] <- NA

# Impute missing values with and without standardization.
ds.std <- nroImpute(data = ds, standard = TRUE)
ds.orig <- nroImpute(data = ds, standard = FALSE)

# Compare against "true" cholesterol values.
rho.std <- cor(ds.std[missing,"CHOL"], dataset[missing,"CHOL"])
rho.orig <- cor(ds.orig[missing,"CHOL"], dataset[missing,"CHOL"])
cat("Correlation, standard = TRUE:  ", rho.std, "\n", sep="")
cat("Correlation, standard = FALSE: ", rho.orig, "\n", sep="")
```

`nroKmeans` *K-means clustering*

Description

K-means clustering for multi-dimensional data.

Usage

```
nroKmeans(data, k = 3, subsample = NULL, balance = 0, message = NULL)
```

Arguments

`data`	A data frame or a matrix.
`k`	Number of centroids.
`subsample`	Number of randomly selected rows used during a single training cycle.
`balance`	Penalty parameter for size difference between clusters.
`message`	If positive, progress information is printed at the specified interval in seconds.

Details

The K centroids are determined by Lloyd's algorithm with Euclidean distances or by using 1 - Pearson correlation as the distance measure.

If `subsample` is less than the number of data rows, a random subset of the specified size is used for each training cycle. By default, `subsample` is set automatically depending on the size of the dataset.

If `balance = 0.0`, the algorithm is applied with no balancing, if `balance = 1.0` all the clusters will be forced to be of equal size. Intermediate values are permitted. Note that if subsampling is applied, balancing may become less accurate.

Value

A list with named elements: `centroids` is a matrix of the main results, `layout` contains the best-matching centroid labels and model residuals for each usable data point and `history` is the chronological record of training errors. The subsampling parameter that was used during training is stored in the element `subsample`.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# Unbalanced K-means clustering.
km0 <- nroKmeans(data = trdata, k = 5, balance = 0.0)
print(table(km0$layout$BMC))
print(km0$centroids)

# Balanced K-means clustering.
km1 <- nroKmeans(data = trdata, k = 5, balance = 1.0)
print(table(km1$layout$BMC))
print(km1$centroids)
```

---

nroKohonen *Self-organizing map*

Description

Interpolates the initial district profiles of a self-organizing map based on pre-determined seed profiles.

Usage

```
nroKohonen(seeds, radius = 3, smoothness = 1.0)
```

Arguments

seeds	A matrix or a data frame of K rows and N columns.
radius	Map radius.
smoothness	Rigidity of the map to adapt to regional differences.

Value

A list of named elements: centroids contains the N-dimensional district profiles, and topology is an H x 6 matrix that contains the 2D spatial layout for the map districts: the first two columns (X, Y) indicate the positions of districts in Cartesian coordinates, the other four columns (RADIUS1, RADIUS2, ANGLE1, ANGLE2) define the perimeter of the district areas for visualisation on a circular map.

Additional parameters are stored as attributes in topology.

The function is named after Teuvo Kohonen, the inventor of the self-organizing map.

See Also

Please see nroKmeans() to create the seeds.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
print(head(sm$centroids))
print(head(sm$topology))
```

---

nroLabel *Label pruning*

Description

Optimize the look and selection of labels on map districts.

Usage

```
nroLabel(topology, values, gap = 2.3)
```

Arguments

topology	A data frame with K rows and six columns, see details.
values	A vector of K values or a K x N data frame, where K is the number of map districts and N is the number of variables.
gap	Minimum distance between map districts with non-empty labels.

Details

The function assigns visible labels for districts based on the absolute deviations from the average district value. The most extreme districts are picked first, and then the remaining districts are prioritized based on their value and distance to the other districts already labeled. Columns that are listed in the attribute "binary" in values are given percentage labels.

Topology can be either the output from nroKohonen() or a data frame in the same format as the element topology within the aforementioned output list.

Value

A matrix with K rows and N columns that contains selected labels for the map districts for each of the columns in values. The output has the attribute 'visible' that contains binary flags to guide visibility.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)

# Assign data points into districts.
matches <- nroMatch(centroids = sm, data = trdata)

# District averages for all variables.
planes <- nroAggregate(topology = sm, districts = matches, data = dataset)

# District labels for cholesterol.
chol <- nroLabel(topology = sm, values = planes[,"CHOL"])
print(head(attr(chol, "visible")))
print(head(chol))

# District labels for all variables.
```

```
colrs <- nroLabel(topology = sm, values = planes)
print(head(attr(colrs, "visible")))
print(head(colrs))
```

---

nroMatch *Best-matching districts*

Description

Compare multi-dimensional data points against the district profiles of a self-organizing map (SOM).

Usage

```
nroMatch(centroids, data)
```

Arguments

centroids	Either a matrix, a data frame or a list that contains the element centroids.
data	A data matrix with identical column names to the centroid matrix.

Details

The input argument centroids can be a matrix or a data frame that contains multivariable data profiles organized row-wise. It can also be the output list object from nroKmeans() or nroTrain().

Value

A vector of integers with elements corresponding to the rows in data. Each element contains the index of the best matching row from centroids.

The vector also has the attribute 'quality' that contains three columns: RESIDUAL is the distance between a point and a centroid in data space (shorter is better), RESIDUAL.z is a scale-independent version of RESIDUAL if the mean residual and standard deviation are available from training history, and COVERAGE shows the proportion of data elements that were available for matching.

The names of the columns that were used for matching are stored in the attribute variables.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata, k = 10)
```

```
# Assign data points into districts.
matches <- nroMatch(centroids = km, data = trdata)
print(head(attr(matches,"quality")))
print(table(matches))
```

---

nroPair *Match similar rows*

Description

Pair up closest matching rows from two datasets

Usage

```
nroPair(data.x, data.y, subsample = 2000, standard = TRUE, priority = 1.0)
```

Arguments

data.x	A matrix or a data frame with column names.
data.y	A matrix or a data frame with column names.
subsample	Maximum number of pairings to test per row.
standard	If TRUE, the scales of variables are standardized for processing.
priority	The proportion of the best matching pairs that are included in the results.

Details

The function detects columns that are shared between the two datasets by their names. Pairs of rows across datasets are then compared using Euclidean distance to determine the best matches.

Value

A data frame that has up to five columns: ROW.x and ROW.y contain the pairings using row indices and DISTANCE contains the distances in (standardized) data space. If row names are available, the columns ROWNAME.x and ROWNAME.y are added.

The output is sorted according to the matching distance and truncated according to the priority parameter.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set row names.
rownames(dataset) <- paste("r", 1:nrow(dataset), sep="")

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# Split by sex.
women <- which(dataset$MALE == 0)
men <- which(dataset$MALE == 1)

# Find the best matches.
pairs <- nroPair(data.x = trdata[women,], data.y = trdata[men,])
print(head(pairs))
```

---

nroPermute *Permutation analysis of map layout*

Description

Estimate the dynamic range and statistical significance for regional patterns on a self-organizing maps using permutations.

Usage

```
nroPermute(map, districts, data, n = 1000, message = NULL)
```

Arguments

map	A list object in the format from nroTrain().
districts	An integer vector of M best matching districts.
data	A numeric vector of M values or an M x N matrix (or data frame), where M is the number of data points and N is the number of variables.
n	Maximum number of permutations per variable.
message	If positive, progress information is printed at the specified interval in seconds.

Details

The input argument map must contain the map topology and the centroid profiles as returned by the functions nroKmeans(), nroKohonen(), or nroTrain().

The input argument districts must contain integers between 1 and K, where K is the number map units. Any other values will be ignored.

Training variables and data points are detected by the column names of map$centroids, the attribute "variables" in districts and the names of elements in districts.

Value

A data frame with eight columns: P.z is a parametric estimate for statistical significance, P.freq is the frequency-based estimate for statistical signicance, and Z is the estimated z-score of how far the observed map plane is from the average randomly generated layout. N.data indicates how many data values were used and N.cycles tells the number of completed permutations. AMPLITUDE is a dynamic range modifier for colors that can be used in `nroColorize()`.

The output also contains the attribute 'zbase' that indicates the normalization factor for the color amplitudes.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set row names.
rownames(dataset) <- paste("r", 1:nrow(dataset), sep="")

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)

# Assign data points into districts.
matches <- nroMatch(centroids = sm, data = trdata)

# Estimate statistics for cholesterol
chol <- nroPermute(map = sm, districts = matches, data = dataset$CHOL)
print(chol[,c("TRAINING", "Z", "P.z", "P.freq")])

# Estimate statistics.
stats <- nroPermute(map = sm, districts = matches, data = dataset)
print(stats[,c("TRAINING", "Z", "P.z", "P.freq")])
```

`nroPlot` *Plot a self-organizing map*

Description

Create a graphical interface for selecting subgroups from multiple map colorings simultaneously.

Usage

```
nroPlot(topology, colors, labels = NULL, subplot = NULL,
        interactive = FALSE, clear = NULL)

nroPlot.save(file, topology, colors, labels = NULL,
             subplot = NULL, font = 1.0)
```

Arguments

`topology`	A data frame with K rows and six or more columns that contain the district positions of a self-organizing map and optional region assignments.
`colors`	A character vector with K elements or a K x N matrix of hexadecimal color codes as strings.
`labels`	A character vector with K elements or a K x N matrix of district labels.
`subplot`	A two-element vector that sets out the number of rows and columns for a grid layout of multiple colorings.
`clear`	If TRUE, all graphics devices are cleared when the plot is refreshed.
`interactive`	If TRUE, an interactive version of the plot is launched.
`file`	If non-empty, the figure is saved as an SVG or HTML file instead of plotting on graphics device.
`font`	Multiplier to adjust font size for SVG and HTML output.

Details

The input `topology` must follow the format from `nroKohonen()`, but may also contain the columns REGION, and REGION.label that specify the names for subsets of districts and the single character labels to be shown on top of those districts. The input can also be the list object as returned by `nroKohonen()`.

The color input can include the attribute 'contrast' that contains a binary vector or a matrix of equal size. If an element is set, it means a dark label or highlight will have better contrast with the background.

The label input can include the attribute 'visible' that contains a binary vector or a matrix of equal size. If an element is set, it means a label is visible, otherwise it will not be shown on the map.

Some non-alphanumeric characters are not supported and will be automatically converted to "_". Too long labels or column names will be truncated.

The default value for `clear` is TRUE to prevent multiple plot windows from accumulating within the RStudio. If you are running R from the terminal or using detached devices, setting `clear` to FALSE will retain the current window when refreshing.

If the file name ends with ".html", an interactive HTML document is produced, otherwise an SVG document is created. We recommend opening the HTML file with a web browser to select regions on large maps (i.e. when the R plot window becomes too slow to use). The HTML page allows you to assign subgroups and to save the results as tab-delimited text.

Value

The main function returns a data frame with K rows that contains the topology and subgrouping information. The .save subroutine returns the number of bytes written in the output file.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Detect binary columns.
dataset <- nroPreprocess(dataset, method = "")

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)

# Assign data points into districts.
matches <- nroMatch(centroids = sm, data = trdata)

# Select a subset of variables and detect binary data.
vars <- c("AGE", "MALE", "uALB", "CHOL", "DIAB_KIDNEY", "DECEASED")
selected <- nroPreprocess(dataset[,vars], method = "")

# Calculate district averages for seleted variables.
vars <- c("AGE", "MALE", "uALB", "CHOL", "DIAB_KIDNEY", "DECEASED")
planes <- nroAggregate(topology = sm, districts = matches, data = selected)

# Estimate statistics.
stats <- nroPermute(map = sm, districts = matches, data = selected)

# Set visuals.
colrs <- nroColorize(values = planes, amplitudes = stats)
labls <- nroLabel(topology = sm, values = planes)

# Add subgrouping information.
topo <- sm$topology
topo$REGION <- ""
topo$REGION[1:8] <- "Center"
topo$REGION[9:21] <- "Perimeter"

# Add subgroup labels.
topo$REGION.label <- ""
topo$REGION.label[1:8] <- "C"
topo$REGION.label[9:21] <- "P"
```

```
# Add subgroup colors.
topo$REGION.color <- ""
topo$REGION.color[1:8] <- "#00f00060"
topo$REGION.color[9:21] <- "#f000f060"

# Plot colorings on screen.
nroPlot(topology = topo, colors = colrs, labels = labls)

# Save colorings in file.
#fn <- "colorings.html"
#n <- nroPlot.save(file = fn, topology = topo,
#    colors = colrs, labels = labls)
#cat(n, " bytes saved in '", fn, "'\n", sep="")
```

`nroPostprocess` *Standardization using existing parameters*

Description

Process a new dataset using a standardization procedure that was created for another dataset

Usage

```
nroPostprocess(data, mapping, reverse = FALSE, trim = FALSE)
```

Arguments

`data`	A matrix or a data frame with column names.
`mapping`	A list object or a matrix or a data frame.
`reverse`	If true, standardized data will be reverted back to original scale.
`trim`	If true, unusable rows and columns are removed.

Details

The input argument can be a data frame with the attribute 'mapping' as returned from `nroPreprocess()` or a list object with the elements `input` and `output` that each contain a data frame or a matrix of equal size.

The function projects the input data to the values in `mapping$input` to determine the positions of the input values with respect to the rows in the model. These positions are then used to interpolate corresponding output values in `mapping$output`.

The mapping elements must have consistent row and column names.

Value

A matrix or data frame of processed values.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Show original data characteristics.
print(summary(dataset))

# Preprocess a subset of data.
ds.pre <- nroPreprocess(dataset[1:100,])
print(summary(ds.pre))

# Repeat preprocessing for the whole dataset (approximation).
ds.post <- nroPostprocess(dataset, ds.pre)
print(summary(ds.post))
```

---

nroPreprocess *Data cleaning and standardization*

Description

Convert to numerical values, remove unusable rows and columns, and standardize scale of each variable.

Usage

```
nroPreprocess(data, method = "standard", clip = 3.0,
    resolution = 100, trim = FALSE)
```

Arguments

data	A matrix or a data frame.
method	Method for standardizing scale and location, see details below.
clip	Range for clipping extreme values in multiples of standard deviations.
resolution	Maximum number of sampling points to capture distribution shape.
trim	if TRUE, empty rows and columns are removed.

Details

Standardization methods include empty string for no action, "standard" for centering by mean and division by standard deviation, "uniform" for normalized ranks between -1 and 1 and "tapered" for a version of the rank-based method that puts more samples around zero.

Clipping is not applied if the method is rank-based.

Value

A matrix of numerical values. A value mapping model is stored in the attribute 'mapping'. The names of binary columns are stored in the attribute 'binary'.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Show original data characteristics.
print(summary(dataset))

# Detect binary columns.
ds <- nroPreprocess(dataset, method = "")
print(attr(ds,"binary"))

# Centering and scaling cholesterol.
ds <- nroPreprocess(dataset$CHOL)
print(summary(ds))

# Centering and scaling.
ds <- nroPreprocess(dataset)
print(summary(ds))

# Tapered ranks.
ds <- nroPreprocess(dataset, method = "tapered")
print(summary(ds))
```

---

nroRcppMatrix *Safety check for Rcpp calls*

Description

Forces all values to numeric to be passed to C++ functions.

Usage

```
nroRcppMatrix(data, trim)
```

Arguments

data	A matrix or a data frame.
trim	if TRUE, empty rows and columns are removed.

Details

Converts all columns to values that have a numeric representation. Detects columns that can be represented as 0s and 1s.

Value

A matrix or a data frame with the attribute 'binary' that contains the names of binary columns and 'excl.rows' and 'excl.columns' contain the names of rows and columns that were excluded.

Author(s)

Ville-Petteri Makinen

Examples

```
# Fully numeric data frame.
x <- data.frame(A=c(1,2,3,4), B=c(0,1,0,NA), C=c(2,3,4,5))
print(nroRcppMatrix(data=x, trim=TRUE))

# Matrix of characters, some of which can be converted to numbers.
x <- matrix(c("1","2","b","4","","6","7","8"), nrow=4, ncol=2)
print(nroRcppMatrix(data=x, trim=TRUE))

# Object that can be converted to numbers.
x <- list(text="abc", value="123")
print(nroRcppMatrix(data=x, trim=TRUE))

# Unusable object.
x <- list(text="abc", value="123", multiple=c("a","b","c"))
print(nroRcppMatrix(data=x, trim=TRUE))
```

`nroRcppVector` *Safety check for Rcpp calls*

Description

Ensures vectors can be passed safely to C++ functions.

Usage

```
nroRcppVector(data, default, numeric = TRUE, empty = TRUE)
```

Arguments

`data`	A vector.
`default`	Default output if input is empty.
`numeric`	If TRUE, output is converted into a numeric vector, otherwise a character vector is returned.
`empty`	If TRUE, empty output is allowed.

Details

Checks the input for size (0 is allowed if empty flag is set) and that it is a vector (or can be converted to a vector).

Value

A numeric or a character vector.

Author(s)

Ville-Petteri Makinen

Examples

```
# Empty input reverts to default.
x <- nroRcppVector(data=NULL, default=NA, empty=TRUE)
print(x)

# Empty input reverts to default, then to specified type.
x <- nroRcppVector(data=NULL, default=123, empty=TRUE, numeric=FALSE)
print(x)

# Convert a logical vector to numbers.
x <- c(TRUE, TRUE, FALSE, TRUE)
names(x) <- c("a","b","c","d")
y <- nroRcppVector(data=x, numeric=TRUE)
print(y)
```

---

nroStatistic *Estimate average value*

Description

Calculate weighted descriptive statistics for datasets with missing and duplicated values.

Usage

```
nroStatistic(data, weights = NULL, method = "center")
```

Arguments

data	A matrix or a data frame.
weights	A matrix or a data frame of positive values with the same row and column names as data.
method	The name of the test statistic, see details.

Details

Methods include "center", "mean", "median", "mode", "min", "max", "range", "sd","iqr", "var" and "number".

For binary columns, the method "center" returns the mean. If there are more than two distinct values (nuniq > 2), the formula rho * mean + (1 - rho) * median, rho = 2.2 / nuniq / log(nuniq + 1) is used. If a column contains duplicated values, the duplicates are expanded linearly prior to calculating the median in the formula. This mitigates numerical artefacts from bootstrapping or permutations.

Range is defined the difference between the maximum and minimum value and "iqr" stands for inter-quartile range.

The method "number" returns the number of usable values.

Value

A numeric vector with elements named according to data columns.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Calculate centers.
mu <- nroStatistic(dataset, method = "center")
print(mu)

# Calculate inter-quartile ranges.
iqr <- nroStatistic(dataset, method = "iqr")
print(iqr)
```

---

nroSummary *Estimate subgroup statistics*

Description

Combine data points that reside in districts that belong to a larger region into a subgroup; compare descriptive statistics between subgroups.

Usage

```
nroSummary(data, districts, regions = NULL, categlim = 8, capacity = 10)
```

Arguments

data	A vector of named M elements or an M x N matrix of data values with row names.

districts	An integer vector of M named elements that indicate the best match out of K districts for each row name in the data matrix, please see nroMatch for an example.
regions	An vector of K elements or a data frame of K rows that defines if a district belongs to a larger region (i.e. a subgroup), see details.
categlim	The threshold for the number of unique values before a variable is considered continuous.
capacity	Maximum number of subgroups to compare.

Details

If defined, the region vector should have K elements where K is the total number of map districts.

The region input can also be a data frame of K rows where the column REGION will be used for assigning district to regions, and REGION.label will be used as the character label as seen on the map, see the output from `nroPlot()` for an example.

Districts and data points are connected by comparing element names in `districts` and names or row names of `data`.

Districts and regions are connected by comparing element values in `districts` and names or row names of `regions`.

If the region vector is empty, each district is automatically assigned to its own region.

Safeguards are in place to prevent crashes from empty categories; this reduces statistical power slightly when numbers are small.

Value

A data frame of summary statistics that contains a row for every combination of subgroups and variables. The chi-squared test is used for comparisons with respect to categorical variables, and rank-regulated t-test and ANOVA are applied to continuous variables. Region labels for each row are stored in the attribute 'labels' and a list that contains the subsets of rows in each region is stored in the attribute 'subgroups'.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Self-organizing map.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata)
```

```
# Assign data points into districts.
matches <- nroMatch(centroids = sm, data = trdata)

# Calculate district averages for urinary albumin.
plane <- nroAggregate(topology = sm, districts = matches,
                      data = dataset$uALB)
plane <- as.vector(plane)

# Assign subgroups based on urinary albumin.
regns <- rep("HighAlb", length.out=length(plane))
regns[which(plane < quantile(plane, 0.67))] <- "MiddleAlb"
regns[which(plane < quantile(plane, 0.33))] <- "LowAlb"

# Add label info and make a data frame.
regns <- data.frame(REGION=regns, REGION.label="",
    stringsAsFactors=FALSE)
regns[which(regns$REGION == "HighAlb"),"REGION.label"] <- "H"
regns[which(regns$REGION == "MiddleAlb"),"REGION.label"] <- "M"
regns[which(regns$REGION == "LowAlb"),"REGION.label"] <- "L"

# Calculate summary statistics.
st <- nroSummary(data = dataset, districts = matches, regions = regns)
print(st[,c("VARIABLE","SUBGROUP","MEAN","P.chisq","P.t","P.anova")])
```

`nroTrain` *Train self-organizing map*

Description

Iterative algorithm to adapt a self-organizing map (SOM) to a set of multivariable data.

Usage

```
nroTrain(map, data, subsample = NULL, message = NULL)
```

Arguments

`map`	A list object as returned by `nroKohonen()`.
`data`	A matrix or a data frame.
`subsample`	Number of rows used during a single training cycle.
`message`	If positive, progress information is printed at the specified interval in seconds.

Details

The map is fitted according to columns that are found both in the SOM centroids and the input data.

If `subsample` is less than the number of data rows, a random subset of the specified size is used for each training cycle. By default, `subsample` is set automatically depending on the size of the dataset.

Value

A copy of the list object `map`, where the element `centroids` is updated according to the data patterns. The quantization errors during training are stored in the element `history`. The subsampling parameter that was used during training is stored in the element `subsample`.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Prepare training data.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- scale.default(dataset[,trvars])

# K-means clustering.
km <- nroKmeans(data = trdata)

# Train with full data.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata, subsample = nrow(trdata))
print(sm$history)

# Train with subsampling.
sm <- nroKohonen(seeds = km)
sm <- nroTrain(map = sm, data = trdata, subsample = 200)
print(sm$history)
```

---

numero.clean *Clean datasets*

Description

Sets row names and removes unusable columns and rows.

Usage

```
numero.clean(..., identity = NULL, na.freq = 0.9,
             num.only = TRUE, select = "")
```

Arguments

`...`	Matrices or a data frames.
`identity`	Name(s) of the column(s) that contain identification information.
`na.freq`	The proportion of how many missing values are allowed in each column and in each row.
`num.only`	If true, only numeric columns are included.
`select`	Indicate if only identities present in all datasets or in exactly one of the datasets are included.

Details

If multiple identity columns are provided, composite identity keys are constructed by concatenating elements from each column with "_" added as a separator.

The frequency of missing values (against /codena.freq) is tested first by column then by row.

Selection can take three values: "" for no selection, "union" for all identities expanded to every dataset, "shared" for only those data points present in all usable datasets or "distinct" for excluding any points that can be found in more than one dataset. Note that the union may result in rows with no usable values.

Value

A data frame if only one input dataset, or a list of data frames if multiple datasets.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Create new versions for testing.
dsA <- dataset[1:250, c("INDEX","AGE","MALE","uALB")]
dsB <- dataset[151:300, c("INDEX","AGE","MALE","uALB","CHOL")]
dsC <- dataset[201:500, c("INDEX","AGE","MALE","DIAB_RETINO")]

# Select all rows.
results <- numero.clean(a = dsA, b = dsB, c = dsC, identity = "INDEX")
cat("\n\nNo selection:\n")
print(nrow(results$a))
print(nrow(results$b))
print(nrow(results$c))

# Select all rows and expanded for all identities.
results <- numero.clean(a = dsA, b = dsB, c = dsC, identity = "INDEX",
                        select = "union")
cat("\n\nUnion:\n")
print(nrow(results$a))
print(nrow(results$b))
print(nrow(results$c))

# Select only rows that are shared between all datasets.
results <- numero.clean(a = dsA, b = dsB, c = dsC, identity = "INDEX",
                        select = "intersection")
cat("\n\nIntersection:\n")
print(nrow(results$a))
print(nrow(results$b))
print(nrow(results$c))
```

```
# Select only rows with a unique INDEX ('dsB' has none).
results <- numero.clean(a = dsA, b = dsB, c = dsC, identity = "INDEX",
                        select = "exclusion")
cat("\n\nExclusion:\n")
print(nrow(results$a))
print(nrow(results$b))
print(nrow(results$c))

# Add extra identification information.
dsA$GROUP <- "A"
dsB$GROUP <- "B"
dsC$GROUP <- "C"

# Select rows with a unique identifier.
results <- numero.clean(a = dsA, b = dsB, c = dsC,
                        identity = c("GROUP","INDEX"),
                        select = "exclusion")
cat("\n\nMulti-identities:\n")
print(nrow(results$a))
print(nrow(results$b))
print(nrow(results$c))
```

`numero.create` *Create a self-organizing map*

Description

Set up a self-organizing map and train it with data

Usage

```
numero.create(data, radius = NULL, smoothness = NULL, subsample = NULL)
```

Arguments

`data`	A matrix or a data frame.
`radius`	Map radius.
`smoothness`	Rigidity of the map to adapt to regional differences.
`subsample`	Number of data points used during a single training cycle.

Details

The parameter `subsample` sets the number of data points that are randomly picked for each training cycle; if the number is substantially less than the size of the dataset, the function will finish quicker.

Value

A list with named elements: `data` contains the training data, `kmeans` is the output from `nroKmeans()` during the initialiation of the SOM, `map` is the finished self-organising map from `nroTrain()` and `layout` contains the output from `nroMatch()` for the training data points.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training set.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)

# Create a self-organizing map.
modl <- numero.create(data = trdata)
```

`numero.evaluate` *Self-organizing map statistics*

Description

Evaluate regional variation of data values on a self-organizing map

Usage

```
numero.evaluate(model, data, ranked = TRUE, n = 1000)
```

Arguments

`model`	A list object that contains a self-organizing map and a data layout.
`data`	A matrix or a data frame.
`ranked`	If true, a rank transform is applied to avoid problems from skewed distributions or outliers.
`n`	Maximum number of permutations per data column.

Details

The input argument `model` can be the output from `numero.create()` or from `numero.quality()`.

Value

A list with named elements: `som` contains the self-organizing map, `layout` contains the district assignments for data points, `planes` contains smoothed district averages from `nroAggregate()`, the element `ranges` contains the reference ranges to be used in `nroColorize()`, the element `statistics` contains the output from `nroPermute()`, the element `palette` is the name of the colormap and the element `data` contains the data points that were used for calculating the statistics.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)

# Create a self-organizing map.
modl <- numero.create(data = trdata)

# Evaluate map statistics.
results <- numero.evaluate(model = modl, data = dataset)
print(results$statistics[,c("TRAINING", "Z", "P.z", "P.freq")])
```

`numero.plot` *Plot results from SOM analysis*

Description

Plot map colorings and save them as vector graphics

Usage

```
numero.plot(results, variables = NULL, topology = NULL, folder = NULL,
            prefix = "figure", reference = NULL, subplot = NULL,
    gain = 1, detach = FALSE, capacity = 500, font = NULL)
```

Arguments

`results`	A list object that contains the self-organizing map and its statistical colorings.
`variables`	A string vector that contains names of variables to show.
`topology`	The topology of a SOM with subgroup labels.
`folder`	Folder path for saving figures.
`prefix`	Prefix for each figure file (if saving enabled).
`reference`	Reference color ranges and scales.
`gain`	Modifier for overall color intensity.
`subplot`	A two-element vector that sets out the number of rows and columns for subplots per figure.
`detach`	Use detached windows for figures.
`capacity`	Maximum number of subplots to show on screen.
`font`	Multiplier to adjust font size for SVG and HTML output.

Details

The input `results` must contain the output from codenumero.evaluate() or similar.

The input argument `topology` can be the topology of a SOM or with additional columns as in the output from `numero.subgroup()`.

The input argument `reference` follows the output format from `numero.evaluate()`.

Possible values for `detach` include "X11", "aqua", TRUE or FALSE. Using multiple figures may result in different behaviour in terminal vs. RStudio instances. The default behaviour is to create detached windows for each figure when the X11 display server is available (e.g. in Linux). To use detached windows in Mac, use the value "aqua". Setting `detach = TRUE` will use a more general approach, however, some systems may behave unpredictably. To create multiple figures that remain docked within the RStudio work window, set `detach = FALSE`.

If a destination folder is provided, all plots are saved in files without plotting them on screen.

Value

The number of figures that were created.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars,
    batch = "MALE", confounders = c("AGE", "T1D_DURAT"))

# Create a self-organizing map.
modl <- numero.create(data = trdata)

# Evaluate map statistics for all variables.
stats <- numero.evaluate(model = modl, data = dataset)

# Plot map colorings.
numero.plot(results = stats)
```

numero.prepare *Prepare datasets for analysis*

Description

Prepare training data by mitigating confounding factors and standardizing values.

Usage

```
numero.prepare(data, variables = NULL, confounders = NULL, batch = NULL,
               method = "standard", pipeline = NULL)
```

Arguments

data	A matrix or a data frame.
variables	A character vector of column names.
confounders	Names of columns that contain confounder data.
batch	The name of the column that contains batch labels.
method	Method to standardize values, see nroPreprocess().
pipeline	Processing parameters from a previous use of the function.

Details

We recommend first applying numero.clean() to the full dataset, then selecting a subset for training using the input argument variables. This preserves any attributes that may be used in Numero functions.

If a previous pipeline is available, it overrides all processing parameters irrespective of other input arguments.

Due to safeguards against numerical instability, the standardized values may deviate slightly from the expected range (<0.1 percent error is typical).

Value

A matrix with the attributes 'pipeline' that contains the processing parameters and 'subsets' that contains row names divided into batches if batch correction was applied.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables using default standardization.
```

```
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)
print(summary(trdata))

# Prepare training values adjusted for age and sex and
# standardized by rank-based method.
trdata <- numero.prepare(data = dataset, variables = trvars,
                         batch = "MALE", confounders = "AGE",
 method = "tapered")
print(summary(trdata))
```

`numero.quality` *Self-organizing map statistics*

Description

Assign new data to map districts and calculate quality measures

Usage

```
numero.quality(model, data = NULL)
```

Arguments

`model`	A list object that contains a self-organizing map and a data layout.
`data`	A matrix or a data frame.

Details

The input argument `model` must be in the the output format as returned by `numero.create()`.

Value

A list with named elements: `som` contains the self-organizing map; `layout` contains the district assignments for data points; `planes` contains smoothed district averages of quality measures, see `nroAggregate()` and `nroMatch()`; the element `ranges` contains the reference ranges to be used in `nroColorize()`; the element `palette` is the name of the colormap to be used for colorings; and `statistics` contains the output from `nroPermute()`.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables.
```

```
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)

# Create a self-organizing map.
modl <- numero.create(data = trdata)

# Analyze map quality.
qc <- numero.quality(model = modl)
```

`numero.subgroup` *Interactive subgroup assignment*

Description

Plot self-organizing map colorings and let the user choose multi-district regions as subgroups

Usage

```
numero.subgroup(results, variables, topology = NULL, reference = NULL,
                gain = 1, detach = FALSE, capacity = 9)
```

Arguments

`results`	A list object that contains the self-organizing map and its statistical colorings.
`variables`	A string vector that contains names of variables to show on screen.
`topology`	A SOM topology or the output from a previous subgrouping session.
`reference`	Reference color ranges and scales.
`gain`	Modifier for overall color intensity.
`detach`	Use a detached window.
`capacity`	Maximum number of subplots to show on screen.

Details

The input `results` must contain the output from codenumero.evaluate() or similar.

The input argument `topology` can be the structure of a SOM or with additional columns as in the output from `nroPlot()`.

The input argument `reference` follows the output format from `numero.evaluate()`.

Setting detach to FALSE will also clear all devices whenever the figure is refreshed. This may be inconvenient when using R from the terminal, for example; please see the help page of `numero.plot()` for using detached window device instead.

If any districts are left unmarked, they are automatically collected into a subgroup of their own.

Value

A data frame similar to the format returned by `nroPlot()`.

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)

# Create a self-organizing map.
modl <- numero.create(data = trdata)

# Evaluate map statistics for all variables.
stats <- numero.evaluate(model = modl, data = dataset)

# Define subgroups, uncomment to launch interactive window.
#elem <- numero.subgroup(results = stats, variables = trvars)
```

`numero.summary` *Summarize subgroup statistics*

Description

Estimates subgroup statistics after self-organizing map analysis

Usage

```
numero.summary(results, topology, data = NULL, capacity = 10)
```

Arguments

`results`	A list object that contains the self-organizing map and its statistical colorings.
`topology`	A SOM topology with additional labels that indicate selected regions.
`data`	A matrix or a data frame.
`capacity`	Maximum number of subgroups to compare.

Details

The input `results` must contain the output from `numero.evaluate()` or similar.

The input argument `topology` must be a definition of a SOM with additional columns as in the output from `numero.subgroup()`.

The function first looks for row names in `data` that are also included in `results`. The rows are then divided into subgroups according to the district assignments in `results` and the region labels in `topology`.

Value

A data frame of summary statistics, see nroSummary() for details. The data frame also contains additional information on which variables were used for the training of the SOM.

The attribute 'layout' is added to the output. It indicates the location on the map and the subgroup name and label for each data row that were included in the analysis.

Author(s)

Ville-Petteri Makinen

Examples

```
# Import data.
fname <- system.file("extdata", "finndiane.txt", package = "Numero")
dataset <- read.delim(file = fname)

# Set identities and manage missing data.
dataset <- numero.clean(dataset, identity = "INDEX")

# Prepare training variables.
trvars <- c("CHOL", "HDL2C", "TG", "CREAT", "uALB")
trdata <- numero.prepare(data = dataset, variables = trvars)

# Create a self-organizing map.
modl <- numero.create(data = trdata)

# Evaluate map statistics for all variables.
stats <- numero.evaluate(model = modl, data = dataset)

# Define subgroups.
x <- stats$planes[,"uALB"]
tops <- which(x >= quantile(x, 0.75, na.rm=TRUE))
bottoms <- which(x <= quantile(x, 0.25, na.rm=TRUE))
elem <- data.frame(stats$map$topology, stringsAsFactors = FALSE)
elem$REGION <- "MiddleAlb"
elem$REGION[tops] <- "HighAlb"
elem$REGION[bottoms] <- "LowAlb"
elem$REGION.label <- "M"
elem$REGION.label[tops] <- "H"
elem$REGION.label[bottoms] <- "L"

# Compare subgroups.
cmp <- numero.summary(results = stats, topology = elem, data = dataset)
```

Index