Package: dtComb (via r-universe)

September 2, 2024

Title Statistical Combination of Diagnostic Tests

Description A system for combining two diagnostic tests using various approaches that include statistical and machine-learning-based methodologies. These approaches are divided into four groups: linear combination methods, non-linear combination methods, mathematical operators, and machine learning algorithms. See the <https://biotools.erciyes.edu.tr/dtComb/> website for more information, documentation, and examples.

Version 1.0.3

URL https://github.com/gokmenzararsiz/dtComb

Language en-US

Depends R (>= 3.5.0)

Imports pROC (>= 1.18.0), caret, epiR, gam, ggplot2, ggpubr, glmnet, OptimalCutpoints

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

NeedsCompilation no

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Author Serra Ilayda Yerlitas [aut, ctb], Serra Bersan Gengec [aut, ctb], Necla Kochan [aut, ctb], Gozde Erturk Zararsiz [aut, ctb], Selcuk Korkmaz [aut, ctb], Gokmen Zararsiz [aut, ctb, cre]

Maintainer Gokmen Zararsiz <gokmen.zararsiz@gmail.com>

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

RemoteUrl https://github.com/gokmenzararsiz/dtcomb

RemoteRef HEAD

RemoteSha 837001bb75398458d87814f168f53ca16275fcff

Contents

`allMethods` *Includes machine learning models used for the mlComb function*

Description

Includes machine learning models used for the mlComb function

Usage

```
data(allMethods)
```

Format

A data frame with 113 rows and 2 variables:

Method Valid name for the function

Model Model name

Examples

```
data(allMethods)
allMethods
```

`availableMethods` *Available classification/regression methods in* `dtComb`

Description

This function returns a data.frame of available classification methods in `dtComb`. These methods are imported from the caret package.

Usage

```
availableMethods()
```

Value

`No return value` contains the method names and explanations of the machine-learning models available for the dtComb package.

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
availableMethods()
```

`dtComb` *dtComb: A Comprehensive R Library for Combining Diagnostic Tests*

Description

The dtComb package calculates combination scores of two biomarkers given under four main categories: linear combinations with the linComb function, non-linear combinations with the nonlinComb function, mathematical operators with the mathComb function, and machine learning algorithms with the mlComb function.

Author(s)

Maintainer: Gokmen Zararsiz <gokmen.zararsiz@gmail.com> [contributor]

Authors:

- Serra Ilayda Yerlitas <ilaydayerlitas340@gmail.com> [contributor]
- Serra Bersan Gengec <serrabersan@gmail.com> [contributor]
- Necla Kochan <necla.kayaalp@gmail.com> [contributor]
- Gozde Erturk Zararsiz <gozdeerturk9@gmail.com> [contributor]
- Selcuk Korkmaz <selcukorkmaz@gmail.com> [contributor]

See Also

Useful links:

- https://github.com/gokmenzararsiz/dtComb

`exampleData1` *Examples data for the dtComb package*

Description

A data set containing the results of diagnostic laparoscopy procedures for 225 patients.

Usage

```
data(exampleData1)
```

Format

A data frame with 225 rows and 3 variables:

group Indicator if the procedure was needed, values needed and not_needed

ddimer Biomarker 1, D-Dimer protein level in blood, ng/mL

log_leukocyte Biomarker 2, Logarithm of Leukocyte count in blood, per mcL

Examples

```
data(exampleData1)
exampleData1$group <- factor(exampleData1$group)
gcol <- c("#E69F00", "#56B4E9")
plot(exampleData1$ddimer, exampleData1$log_leukocyte,
  col = gcol[as.numeric(exampleData1$group)]
)
```

exampleData2 *A data set containing the carriers of a rare genetic disorder for 120 samples.*

Description

A data set containing the carriers of a rare genetic disorder for 120 samples.

Usage

```
data(exampleData2)
```

Format

A data frame with 120 rows and 5 variables:

Group Indicator if the person was carriers, values carriers and normals

m1 Biomarker 1, 1. measurement blood sample

m2 Biomarker 2, 2. measurement blood sample

m3 Biomarker 3, 3. measurement blood sample

m4 Biomarker 4, 4. measurement blood sample

Examples

```
data(exampleData2)
exampleData2$Group <- factor(exampleData2$Group)
gcol <- c("#E69F00", "#56B4E9")
plot(exampleData2$m1, exampleData2$m2,
  col = gcol[as.numeric(exampleData2$Group)]
)
```

exampleData3 *A simulation data containing 250 diseased and 250 healthy individuals.*

Description

A simulation data containing 250 diseased and 250 healthy individuals.

Usage

```
data(exampleData3)
```

Format

A data frame with 500 rows and 3 variables:

status Indicator of one's condition, values healthy and diseased

marker1 1. biomarker

marker2 2. biomarker

Examples

```
data(exampleData3)
exampleData3$status <- factor(exampleData3$status)
gcol <- c("#E69F00", "#56B4E9")
plot(exampleData3$marker1, exampleData3$marker2,
  col = gcol[as.numeric(exampleData3$status)]
)
```

---

helper_minimax *Helper function for minimax method.*

Description

The helper_minimax function calculates the combination coefficient and optimized value of given biomarkers for the minimax method.

Usage

```
helper_minimax(t, neg.set, pos.set, markers, status)
```

Arguments

t	a numeric parameter that will be estimated in minimax method for the combination score
neg.set	a numeric data frame that contains the observation with negative status
pos.set	a numeric data frame that contains the observation with positive status
markers	a numeric data frame that contains the biomarkers
status	a factor data frame that includes the actual disease status of the patients

Value

A numeric Optimized value calculated with combination scores using t

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- cbind(exampleData1$ddimer, exampleData1$log_leukocyte)
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))

neg.set <- markers[status == levels(status)[1], ]
pos.set <- markers[status == levels(status)[2], ]

t <- 0.5

stat <- helper_minimax(t,
  neg.set = neg.set, pos.set = pos.set,
  markers = markers, status
)
```

`helper_minmax` *Helper function for minmax method.*

Description

The `helper_minmax` function estimates optimized value of given biomarkers for the minmax method.

Usage

```
helper_minmax(lambda, neg.set, pos.set)
```

Arguments

`lambda`	a `numeric` parameter that will be estimated in minmax method for the combination score
`neg.set`	a `numeric` data frame that contains the observations with negative status
`pos.set`	a `numeric` data frame that contains the observations with positive status

Value

A `numeric` value for the estimated optimized value

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- cbind(exampleData1$ddimer, exampleData1$log_leukocyte)
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))

neg.set <- markers[status == levels(status)[1], ]
pos.set <- markers[status == levels(status)[2], ]

lambda <- 0.5

stat <- helper_minmax(lambda, neg.set = neg.set, pos.set = pos.set)
```

`helper_PCL` *Helper function for PCL method.*

Description

The `helper_PCL` function estimates the optimized value of given biomarkers for the PCL method.

Usage

```
helper_PCL(lambda, neg.set, pos.set)
```

Arguments

`lambda`	a `numeric` parameter that will be estimated in minmax method for the combination score
`neg.set`	a `numeric` data frame that contains the observation with negative status
`pos.set`	a `numeric` data frame that contains the observation with positive status

Value

A `numeric` value for the estimated optimized value

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- cbind(exampleData1$ddimer, exampleData1$log_leukocyte)
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))

neg.set <- markers[status == levels(status)[1], ]
pos.set <- markers[status == levels(status)[2], ]

lambda <- 0.5

stat <- helper_PCL(lambda, neg.set = neg.set, pos.set = pos.set)
```

`helper_PT` *Helper function for PT method.*

Description

The `helper_PT` function estimates the optimized value of given biomarkers for the PT method.

Usage

```
helper_PT(lambda, neg.set, pos.set)
```

Arguments

`lambda`	a `numeric` parameter that will be estimated in minmax method for the combination score
`neg.set`	a `numeric` data frame that contains the observation with negative status
`pos.set`	a `numeric` data frame that contains the observation with positive status

Value

A `numeric` value for the estimated optimized value

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- cbind(exampleData1$ddimer, exampleData1$log_leukocyte)
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))

neg.set <- markers[status == levels(status)[1], ]
pos.set <- markers[status == levels(status)[2], ]

lambda <- 0.5

stat <- helper_PT(lambda, neg.set = neg.set, pos.set = pos.set)
```

`helper_TS` *Helper function for TS method.*

Description

The `helper_TS` function calculates the combination coefficient and optimized value of given biomarkers for the TS method.

Usage

```
helper_TS(theta, markers, status)
```

Arguments

`theta`	a `numeric` parameter that will be estimated in TS method for the combination score
`markers`	a `numeric` data frame that contains the biomarkers
`status`	a `factor` data frame that includes the actual disease status of the patients

Value

A `numeric` Optimized value calculated with combination scores using theta

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- cbind(exampleData1$ddimer, exampleData1$log_leukocyte)
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))

t <- 0.5

stat <- helper_TS(theta = t, markers = markers, status = status)
```

---

kappa.accuracy *Calculate Cohen's kappa and accuracy.*

Description

The kappa.accuracy calculates Cohen's kappa and accuracy.

Usage

```
## S3 method for class 'accuracy'
kappa(DiagStatCombined)
```

Arguments

DiagStatCombined
: a numeric table of confusion matrix of the calculated combination score.

Value

A list of Cohen's kappa and accuracy values

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

linComb *Combine two diagnostic tests with several linear combination methods.*

Description

The linComb function calculates the combination scores of two diagnostic tests selected among several linear combination methods and standardization options.

Usage

```
linComb(
  markers = NULL,
  status = NULL,
  event = NULL,
 method = c("scoring", "SL", "logistic", "minmax", "PT", "PCL", "minimax", "TS"),
  resample = c("none", "cv", "repeatedcv", "boot"),
  nfolds = 5,
  nrepeats = 3,
  niters = 10,
  standardize = c("none", "range", "zScore", "tScore", "mean", "deviance"),
  ndigits = 0,
  show.plot = TRUE,
  direction = c("auto", "<", ">"),
  conf.level = 0.95,
 cutoff.method = c("CB", "MCT", "MinValueSp", "MinValueSe", "ValueSp", "ValueSe",
  "MinValueSpSe", "MaxSp", "MaxSe", "MaxSpSe", "MaxProdSpSe", "ROC01", "SpEqualSe",
    "Youden", "MaxEfficiency", "Minimax", "MaxDOR", "MaxKappa", "MinValueNPV",
  "MinValuePPV", "ValueNPV", "ValuePPV", "MinValueNPVPPV", "PROC01", "NPVEqualPPV",
    "MaxNPVPPV", "MaxSumNPVPPV", "MaxProdNPVPPV", "ValueDLR.Negative",
  "ValueDLR.Positive", "MinPvalue", "ObservedPrev", "MeanPrev", "PrevalenceMatching"),
  ...
)
```

Arguments

markers	a numeric a numeric data frame that includes two diagnostic tests results
status	a factor vector that includes the actual disease status of the patients
event	a character string that indicates the event in the status to be considered as positive event
method	a character string specifying the method used for combining the markers. **Notations:** Before getting into these methods, let us first introduce some notations that will be used throughout this vignette. Let $D_i, i = 1, 2, \ldots, n_1$ be the marker values of ith individual in diseased group, where $D_i = (D_{i1}, D_{i2})$ and $H_j, j = 1, 2, \ldots, n_2$ be the marker values of jth individual in healthy group, where $H_j = H_{j1}, H_{j2}$. Let $x_i 1 = c(D_{i1}, H_{j1})$ be the values of the first marker,

and $x_i2 = c(D_{i2}, H_{j2})$ be values of the second marker for the ith individual $i = 1, 2, \ldots, n$. Let $D_{i,min} = min(D_{i1}, D_{i2}), D_{i,max} = max(D_{i1}, D_{i2}), H_{j,min} = min(H_{j1}, H_{j2}), H_{j,max} = max(H_{j1}, H_{j2})$ and c_i be be the resulting combination score for the ith individual.

The available methods are:

- **Logistic Regression** (`logistic`): Combination score obtained by fitting a logistic regression modelis as follows:

$$c_i = \left(\frac{e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}}}{1 + e^{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}}} \right)$$

  A combination score obtained by fitting a logistic regression model typically refers to the predicted probability or score assigned to each observation in a dataset based on the logistic regression model's fitted values

- **Scoring based on Logistic Regression** (`scoring`): Combination score is obtained using the slope values of the relevant logistic regression model, slope values are rounded to the number of digits taken from the user.

$$c_i = \beta_1 x_{i1} + \beta_2 x_{i2}$$

- **Pepe & Thompson's method** (`PT`): The Pepe and Thompson combination score, developed using their optimal linear combination technique, aims to maximize the Mann-Whitney statistic in the same way that the Min-max method does. Unlike the Min-max method, the Pepe and Thomson method takes into account all marker values instead of just the lowest and maximum values.

$$maximize\ U(\alpha) = \left(\frac{1}{n_1, n_2} \right) \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} I(D_{i1} + \alpha D_{i2} >= H_{j1} + \alpha H_{j2})$$

$$c_i = x_{i1} + \alpha x_{i2}$$

- **Pepe, Cai & Langton's method** (`PCL`): Pepe, Cai and Langton combination score obtained by using AUC as the parameter of a logistic regression model.

$$maximize\ U(\alpha) = \left(\frac{1}{n_1, n_2} \right) \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} I(D_{i1} + \alpha D_{i2} >$$

$$H_{j1} + \alpha H_{j2}) + \left(\frac{1}{2} \right) I(D_{i1} + \alpha D_{i2} = H_{j1} + \alpha H_{j2})$$

- **Min-Max method** (`minmax`): This method linearly combines the minimum and maximum values of the markers by finding a parameter,α , that maximizes the Mann-Whitney statistic, an empirical estimate of the ROC area.

$$maximize\ U(\alpha) = \left(\frac{1}{n_1, n_2} \right) \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} I(D_{i,max} + \alpha D_{i,min} > H_{j,max} + \alpha H_{j,min})$$

$$c_i = x_{i,max} + \alpha x_{i,min}$$

where $x_{i,max} = max(x_{i1}, x_{i2})$ and $x_{i,min} = min(x_{i1}, x_{i2})$

- **Su & Liu's method** (`SL`): The Su and Liu combination score is computed through Fisher's discriminant coefficients, which assumes that the underlying data follow a multivariate normal distribution, and the covariance matrices across different classes are assumed to be proportional.Assuming that $D \sim N(\mu_D, \sum_D)$ and $H \sim N(\mu_H, \sum_H)$ represent the multivariate normal distributions for the diseased and non-diseased groups, respectively. The Fisher's coefficients are as follows:

$$(\alpha, \beta) = (\sum_D + \sum_H)^{-1} \mu$$

where$\mu_= \mu_D - \mu_H$.The combination score in this case is:

$$c_i = \alpha x_{i1} + \beta x_{i2}$$

- **Minimax approach** (`minimax`): Combination score obtained with the Minimax procedure; t parameter is chosen as the value that gives the maximum AUC from the combination score. Suppose that D follows a multivariate normal distribution $D \sim N(\mu_D, \sum_D)$, representing diseased group and H follows a multivariate normal distribution $H \sim N(\mu_H, \sum_H)$, representing the non-diseased group. Then Fisher's coefficients are as follows:

$$(\alpha, \beta) = [t\sum_D + (1-t)\sum_H]^{-1}(\mu_D - \mu_H)$$

$$c_i = b_1 x_1 + b_2 x_2$$

- **Todor & Saplacan's method** (`TS`):Combination score obtained by using the trigonometric functions of the Θ value that optimizes the corresponding AUC.

$$c_i = sin(\theta) x_{i1} + cos(\theta) x_{i2}$$

`resample` a `character` string indicating the name of the resampling options. Bootstrapping Cross-validation and repeated cross-validation are given as the options for resampling, along with the number of folds and number of repeats.

- `boot`: Bootstrapping is performed similarly; the dataset is divided into folds with replacement and models are trained and tested in these folds to determine the best parameters for the given method and dataset.
- `cv`: Cross-validation resampling, the dataset is divided into the number of folds given without replacement; in each iteration, one fold is selected as the test set, and the model is built using the remaining folds and tested on the test set. The corresponding AUC values and the parameters used for the combination are kept in a list. The best-performed model is selected, and the combination score is returned for the whole dataset.
- `repeatedcv`: Repeated cross-validation the process is repeated, and the best-performed models selected at each step are stored in another list; the best performed among these models is selected to be applied to the entire dataset.

`nfolds` a `numeric` value that indicates the number of folds for cross validation based resampling methods (5, default)

`nrepeats` a `numeric` value that indicates the number of repeats for "repeatedcv" option of resampling methods (3, default)

`niters` a `numeric` value that indicates the number of bootstrapped resampling iterations (10, default)

`standardize` a `character` string indicating the name of the standardization method. The default option is no standardization applied. Available options are:

- **Z-score** (`zScore`): This method scales the data to have a mean of 0 and a standard deviation of 1. It subtracts the mean and divides by the standard deviation for each feature. Mathematically,

$$Z - score = \frac{x - (\overline{x})}{sd(x)}$$

  where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.
- **T-score** (`tScore`): T-score is commonly used in data analysis to transform raw scores into a standardized form. The standard formula for converting a raw score x into a T-score is:

$$T - score = \left(\frac{x - (\overline{x})}{sd(x)} \times 10\right) + 50$$

  where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.
- **Range (a.k.a. min-max scaling)** (`range`): This method transforms data to a specific range, between 0 and 1. The formula for this method is:

$$Range = \frac{x - min(x)}{max(x) - min(x)}$$

- **Mean** (`mean`): This method, which helps to understand the relative size of a single observation concerning the mean of dataset, calculates the ratio of each data point to the mean value of the dataset.

$$Mean = \frac{x}{\overline{x}}$$

  where x is the value of a marker and $\overline{x}$ is the mean of the marker.
- **Deviance** (`deviance`): This method, which allows for comparison of individual data points in relation to the overall spread of the data, calculates the ratio of each data point to the standard deviation of the dataset.

$$Deviance = \frac{x}{sd(x)}$$

  where x is the value of a marker and $sd(x)$ is the standard deviation of the marker.

ndigits	a integer value to indicate the number of decimal places to be used for rounding in Scoring method (0, default)
show.plot	a logical a logical. If TRUE, a ROC curve is plotted. Default is TRUE
direction	a character string determines in which direction the comparison will be made. ">": if the predictor values for the control group are higher than the values of the case group (controls > cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < cases).
conf.level	a numeric values determines the confidence interval for the roc curve(0.95, default).
cutoff.method	a character string determines the cutoff method for the roc curve.
...	further arguments. Currently has no effect on the results.

Value

A list of numeric linear combination scores calculated according to the given method and standardization option.

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- exampleData1[, -1]
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))
event <- "needed"

score1 <- linComb(
  markers = markers, status = status, event = event,
  method = "logistic", resample = "none", show.plot = TRUE,
  standardize = "none", direction = "<", cutoff.method = "Youden"
)

# call data
data(exampleData2)

# define the function parameters
markers <- exampleData2[, -c(1:3, 6:7)]
status <- factor(exampleData2$Group, levels = c("normals", "carriers"))
event <- "carriers"

score2 <- linComb(
  markers = markers, status = status, event = event,
  method = "PT", resample = "none", standardize = "none", direction = "<",
  cutoff.method = "Youden"
```

```
)

score3 <- linComb(
  markers = markers, status = status, event = event,
  method = "minmax", resample = "none", direction = "<",
  cutoff.method = "Youden"
)
```

`mathComb` *Combine two diagnostic tests with several mathematical operators and distance measures.*

Description

The `mathComb` function returns the combination results of two diagnostic tests with different mathematical operators, distance measures, standardization, and transform options.

Usage

```
mathComb(
  markers = NULL,
  status = NULL,
  event = NULL,
  method = c("add", "multiply", "divide", "subtract", "distance", "baseinexp",
    "expinbase"),
 distance = c("euclidean", "manhattan", "chebyshev", "kulczynski_d", "lorentzian",
    "avg", "taneja", "kumar-johnson"),
  standardize = c("none", "range", "zScore", "tScore", "mean", "deviance"),
  transform = c("none", "log", "exp", "sin", "cos"),
  show.plot = TRUE,
  direction = c("auto", "<", ">"),
  conf.level = 0.95,
 cutoff.method = c("CB", "MCT", "MinValueSp", "MinValueSe", "ValueSp", "ValueSe",
  "MinValueSpSe", "MaxSp", "MaxSe", "MaxSpSe", "MaxProdSpSe", "ROC01", "SpEqualSe",
    "Youden", "MaxEfficiency", "Minimax", "MaxDOR", "MaxKappa", "MinValueNPV",
  "MinValuePPV", "ValueNPV", "ValuePPV", "MinValueNPVPPV", "PROC01", "NPVEqualPPV",
    "MaxNPVPPV", "MaxSumNPVPPV", "MaxProdNPVPPV", "ValueDLR.Negative",
  "ValueDLR.Positive", "MinPvalue", "ObservedPrev", "MeanPrev", "PrevalenceMatching"),
  ...
)
```

Arguments

`markers`	a `numeric` data frame that includes two diagnostic tests results
`status`	a `factor` vector that includes the actual disease status of the patients
`event`	a `character` string that indicates the event in the status to be considered as positive event

`method` a `character` string specifying the method used for combining the markers. The available methods are:

- `add`: Combination score obtained by adding markers
- `multiply`: Combination score obtained by multiplying markers
- `divide`: Combination score obtained by dividing markers
- `subtract`: Combination score obtained by subtracting markers
- `distance`: Combination score obtained with the help of distance measures.
- `baseinexp`: Combination score obtained by marker1 power marker2.
- `expinbase`: Combination score obtained by marker2 power marker1.

`distance` a `character` string specifying the method used for combining the markers. The available methods are:

- **Euclidean** (`euclidean`): $c_i = \sqrt{(x_{i1} - 0)^2 + (x_{i2} - 0)^2}$
- **Manhattan**(`manhattan`): $c_i = |x_{i1} - 0| + |x_{i2} - 0|$
- **Chebyshev** (`chebyshev`): $c_i = max|x_{i1} - 0|, |x_{i2} - 0|$
- **Kulczynski** (`kulczynski_d`): $c_i = \frac{|x_{i1}-0|+|x_{i2}-0|}{min(x_{i1},x_{i2})}$
- **Lorentzian** (`lorentzian`): $c_i = (ln(1 + |x_{i1} - 0|)) + (ln(1 + |x_{i2} - 0|))$
- **Taneja** (`taneja`): $c_i = z_1 \times \left(log \frac{z_1}{\sqrt{(x_{i1} \times \epsilon)}} \right) + z_2 \times \left(log \frac{z_2}{\sqrt{(x_{i2} \times \epsilon)}} \right)$
- **Kumar-Johnson** (`kumar-johnson`): $c_i = \frac{(x_{i1}-0)^2}{2(x_{i1} \times \epsilon)} + \frac{(x_{i2}-0)^2}{2(x_{i2} \times \epsilon)}, \epsilon = 0.00001$
- **Avg** (`avg`):

$$(L_1, L_n) = \frac{|x_{i1} - 0| + |x_{i2} - 0| + max(x_{i1} - 0), (x_{i2} - 0)}{2}$$

`standardize` a `character` string indicating the name of the standardization method. The default option is no standardization applied. Available options are:

- **Z-score** (`zScore`): This method scales the data to have a mean of 0 and a standard deviation of 1. It subtracts the mean and divides by the standard deviation for each feature. Mathematically,

$$Z - score = \frac{x - (\overline{x})}{sd(x)}$$

  where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **T-score** (`tScore`): T-score is commonly used in data analysis to transform raw scores into a standardized form. The standard formula for converting a raw score x into a T-score is:

$$T - score = \left(\frac{x - (\overline{x})}{sd(x)} \times 10 \right) + 50$$

  where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **Range (a.k.a. min-max scaling)** (`range`): This method transforms data to a specific range, between 0 and 1. The formula for this method is:

$$Range = \frac{x - min(x)}{max(x) - min(x)}$$

- **Mean** (`mean`): This method, which helps to understand the relative size of a single observation concerning the mean of dataset, calculates the ratio of each data point to the mean value of the dataset.

$$Mean = \frac{x}{\overline{x}}$$

where x is the value of a marker and $\overline{x}$ is the mean of the marker.

- **Deviance** (`deviance`): This method, which allows for comparison of individual data points in relation to the overall spread of the data, calculates the ratio of each data point to the standard deviation of the dataset.

$$Deviance = \frac{x}{sd(x)}$$

where x is the value of a marker and $sd(x)$ is the standard deviation of the marker.

`transform` a `character` string indicating the name of the standardization method. The default option is no standardization applied. Available options are:

- `log`: Applies logarithm transform to markers before calculating combination score
- `exp`: Applies exponential transform to markers before calculating combination score
- `sin`: Applies sinus trigonometric transform to markers before calculatin combination score
- `cos`: Applies cosinus trigonometric transform to markers before calculating combination score

`show.plot` a `logical` a `logical`. If TRUE, a ROC curve is plotted. Default is TRUE

`direction` a `character` string determines in which direction the comparison will be made. ">": if the predictor values for the control group are higher than the values of the case group (controls > cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < cases).

`conf.level` a `numeric` values determines the confidence interval for the roc curve(0.95, default).

`cutoff.method` a `character` string determines the cutoff method for the roc curve.

`...` further arguments. Currently has no effect on the results.

Value

A list of `numeric` mathematical combination scores calculated according to the given method and standardization option

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
data(exampleData1)
markers <- exampleData1[, -1]
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))
event <- "needed"
direction <- "<"
cutoff.method <- "Youden"

score1 <- mathComb(
  markers = markers, status = status, event = event,
  method = "distance", distance = "avg", direction = direction, show.plot = FALSE,
  standardize = "none", cutoff.method = cutoff.method
)

score2 <- mathComb(
  markers = markers, status = status, event = event,
  method = "baseinexp", transform = "exp", direction = direction,
  cutoff.method = cutoff.method
)

score3 <- mathComb(
  markers = markers, status = status, event = event,
  method = "subtract", direction = "auto", cutoff.method = "MinValueSp", transform = "sin"
)
```

`mlComb` *Combine two diagnostic tests with Machine Learning Algorithms.*

Description

The `mlComb` function calculates the combination scores of two diagnostic tests selected among several Machine Learning Algorithms

Usage

```
mlComb(
  markers = NULL,
  status = NULL,
  event = NULL,
  method = NULL,
  resample = NULL,
  niters = 5,
  nfolds = 5,
```

```
  nrepeats = 3,
  preProcess = NULL,
  show.plot = TRUE,
  B = 25,
  direction = c("auto", "<", ">"),
  conf.level = 0.95,
 cutoff.method = c("CB", "MCT", "MinValueSp", "MinValueSe", "ValueSp", "ValueSe",
  "MinValueSpSe", "MaxSp", "MaxSe", "MaxSpSe", "MaxProdSpSe", "ROC01", "SpEqualSe",
    "Youden", "MaxEfficiency", "Minimax", "MaxDOR", "MaxKappa", "MinValueNPV",
  "MinValuePPV", "ValueNPV", "ValuePPV", "MinValueNPVPPV", "PROC01", "NPVEqualPPV",
    "MaxNPVPPV", "MaxSumNPVPPV", "MaxProdNPVPPV", "ValueDLR.Negative",
  "ValueDLR.Positive", "MinPvalue", "ObservedPrev", "MeanPrev", "PrevalenceMatching"),
  ...
)
```

Arguments

markers	a numeric data frame that includes two diagnostic tests results
status	a factor vector that includes the actual disease status of the patients
event	a character string that indicates the event in the status to be considered as positive event
method	a character string specifying the method used for combining the markers. For the available methods see availableMethods() **IMPORTANT**: See https://topepo.github.io/caret/available-models.html for further information about the methods used in this function.
resample	a character string that indicates the resampling method used while training the model. The available methods are "boot", "boot632", "optimism_boot", "boot_all", "cv", "repeatedcv", "LOOCV", "LGOCV", "none", "oob", "adaptive_cv", "adaptive_boot" and "adaptive_LGOCV". for details of these resampling methods see ?caret::trainControl
niters	a numeric value that indicates the number of bootstrapped resampling iterations (10, default)
nfolds	a numeric value that indicates the number of folds for cross validation based resampling methods (5, default)
nrepeats	a numeric value that indicates the number of repeats for "repeatedcv" option of resampling methods (3, default)
preProcess	a character string that indicates the pre-processing options to be applied in the data before training the model. Available pre-processing methods are: "BoxCox", "YeoJohnson", "expoTrans", "center", "scale", "range", "knnImpute", "bagImpute", "medianImpute", "pca", "ica", "spatialSign", "corr", "zv", "nzv", and "conditionalX". For detailed information about the methods see ?caret::preProcess
show.plot	a logical a logical. If TRUE, a ROC curve is plotted. Default is TRUE
B	a numeric value that is the number of bootstrap samples for bagging classifiers, "bagFDA", "bagFDAGCV", "bagEarth" and "bagEarthGCV". (25, default)

direction	a character string determines in which direction the comparison will be made. ">": if the predictor values for the control group are higher than the values of the case group (controls > cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < cases).
conf.level	a numeric value to determine the confidence interval for the ROC curve(0.95, default).
cutoff.method	a character string determines the cutoff method for the ROC curve.
...	optional arguments passed to selected classifiers.

Value

A list of AUC values, diagnostic statistics, coordinates of the ROC curve for the combination score obtained using Machine Learning Algorithms as well as the given biomarkers individually, a comparison table for the AUC values of individual biomarkers and combination score obtained and the fitted model.

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- exampleData1[, -1]
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))
event <- "needed"

model <- mlComb(
  markers = markers, status = status, event = event,
  method = "knn", resample = "repeatedcv", nfolds = 10, nrepeats = 5,
  preProcess = c("center", "scale"), direction = "<", cutoff.method = "Youden"
)
```

---

nonlinComb *Combine two diagnostic tests with several non-linear combination methods.*

Description

The nonlinComb function calculates the combination scores of two diagnostic tests selected among several non-linear combination methods and standardization options

Usage

```
nonlinComb(
  markers = NULL,
  status = NULL,
  event = NULL,
 method = c("polyreg", "ridgereg", "lassoreg", "elasticreg", "splines", "sgam", "nsgam"),
  degree1 = 3,
  degree2 = 3,
  df1 = 4,
  df2 = 4,
  resample = c("none", "cv", "repeatedcv", "boot"),
  nfolds = 5,
  nrepeats = 3,
  niters = 10,
  standardize = c("none", "range", "zScore", "tScore", "mean", "deviance"),
  include.interact = FALSE,
  alpha = 0.5,
  show.plot = TRUE,
  direction = c("auto", "<", ">"),
  conf.level = 0.95,
 cutoff.method = c("CB", "MCT", "MinValueSp", "MinValueSe", "ValueSp", "ValueSe",
  "MinValueSpSe", "MaxSp", "MaxSe", "MaxSpSe", "MaxProdSpSe", "ROC01", "SpEqualSe",
    "Youden", "MaxEfficiency", "Minimax", "MaxDOR", "MaxKappa", "MinValueNPV",
  "MinValuePPV", "ValueNPV", "ValuePPV", "MinValueNPVPPV", "PROC01", "NPVEqualPPV",
    "MaxNPVPPV", "MaxSumNPVPPV", "MaxProdNPVPPV", "ValueDLR.Negative",
  "ValueDLR.Positive", "MinPvalue", "ObservedPrev", "MeanPrev", "PrevalenceMatching"),
  ...
)
```

Arguments

markers	a numeric data frame that includes two diagnostic tests results
status	a factor vector that includes the actual disease status of the patients
event	a character string that indicates the event in the status to be considered as positive event
method	a character string specifying the method used for combining the markers. The available methods are:

- **Logistic Regression with Polynomial Feature Space** (polyreg): The method builds a logistic regression model with the polynomial feature space and returns the probability of a positive event for each observation.
- **Ridge Regression with Polynomial Feature Space** (ridgereg): Ridge regression is a shrinkage method used to estimate the coefficients of highly correlated variables and in this case the polynomial feature space created from two markers. For the implementation of the method, glmnet() library is used with two functions: cv.glmnet() to run a cross validation model to determine the tuning parameter λ and glmnet() to fit the model with the selected tuning parameter. For Ridge regression, the glmnet() package is

integrated into the dtComb package to facilitate the implementation of this method.

- **Lasso Regression with Polynomial Feature Space** (`lassoreg`): Lasso regression, like Ridge regression, is a type of shrinkage method. However, a notable difference is that Lasso tends to set some feature coefficients to zero, making it useful for feature elimination. It also employs cross-validation for parameter selection and model fitting using the glmnet library.
- **Elastic Net Regression with Polynomial Feature Space** (`elasticreg`): Elastic Net regression is a hybrid model that merges the penalties from Ridge and Lasso regression, aiming to leverage the strengths of both approaches. This model involves two parameters: λ, similar to Ridge and Lasso, and α, a user-defined mixing parameter ranging between 0 (representing Ridge) and 1 (representing Lasso). The α parameter determines the balance or weights between the loss functions of Ridge and Lasso regressions.
- **Splines** (`splines`): Another non-linear approach to combine markers involves employing regression models within a polynomial feature space. This approach applies multiple regression models to the dataset using a function derived from piecewise polynomials. This implementation uses splines with user-defined degrees of freedom and degrees for the fitted polynomials. The splines library is employed to construct piecewise logistic regression models using base splines.
- **Generalized Additive Models with Smoothing Splines and Generalized Additive Models with Natural Cubic Splines** (`sgam` & `nsgam`): In addition to the basic spline structure, Generalized Additive Models are applied with natural cubic splines and smoothing splines using the gam library in R.

`degree1` a `numeric` value for polynomial based methods indicates the degree of the feature space created for marker 1, for spline based methods the degree of the fitted polynomial between each node for marker 1. (3, default)

`degree2` a `numeric` value for polynomial based methods indicates the degree of the feature space created for marker 2, for spline based methods the degree of the fitted polynomial between each node for marker 2 (3, default)

`df1` a `numeric` value that indicates the number of knots as the degrees of freedom in spline based methods for marker 1 (4, default)

`df2` a `numeric` value that indicates the number of knots as the degrees of freedom in spline based methods for marker 2 (4, default)

`resample` a `character` string indicating the name of the resampling options. Bootstrapping Cross-validation and repeated cross-validation are given as the options for resampling, along with the number of folds and number of repeats.

- `boot`: Bootstrapping is performed similarly; the dataset is divided into folds with replacement and models are trained and tested in these folds to determine the best parameters for the given method and dataset.
- `cv`: Cross-validation resampling, the dataset is divided into the number of folds given without replacement; in each iteration, one fold is selected as

the test set, and the model is built using the remaining folds and tested on the test set. The corresponding AUC values and the parameters used for the combination are kept in a list. The best-performed model is selected, and the combination score is returned for the whole dataset.

- `repeatedcv`: Repeated cross-validation the process is repeated, and the best-performed models selected at each step are stored in another list; the best performed among these models is selected to be applied to the entire dataset.

`nfolds` a `numeric` value that indicates the number of folds for cross validation based resampling methods (5, default)

`nrepeats` a `numeric` value that indicates the number of repeats for "repeatedcv" option of resampling methods (3, default)

`niters` a `numeric` value that indicates the number of bootstrapped resampling iterations (10, default)

`standardize` a `character` string indicating the name of the standardization method. The default option is no standardization applied. Available options are:

- **Z-score** (`zScore`): This method scales the data to have a mean of 0 and a standard deviation of 1. It subtracts the mean and divides by the standard deviation for each feature. Mathematically,

$$Z - score = \frac{x - (\overline{x})}{sd(x)}$$

where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **T-score** (`tScore`): T-score is commonly used in data analysis to transform raw scores into a standardized form. The standard formula for converting a raw score x into a T-score is:

$$T - score = \left(\frac{x - (\overline{x})}{sd(x)} \times 10\right) + 50$$

where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **Range (a.k.a. min-max scaling)** (`range`): This method transforms data to a specific range, between 0 and 1. The formula for this method is:

$$Range = \frac{x - min(x)}{max(x) - min(x)}$$

- **Mean** (`mean`): This method, which helps to understand the relative size of a single observation concerning the mean of dataset, calculates the ratio of each data point to the mean value of the dataset.

$$Mean = \frac{x}{\overline{x}}$$

where x is the value of a marker and $\overline{x}$ is the mean of the marker.

- **Deviance** (`deviance`): This method, which allows for comparison of individual data points in relation to the overall spread of the data, calculates the ratio of each data point to the standard deviation of the dataset.

$$Deviance = \frac{x}{sd(x)}$$

where x is the value of a marker and $sd(x)$ is the standard deviation of the marker.

`include.interact` a `logical` indicator that specifies whether to include the interaction between the markers to the feature space created for polynomial based methods (FALSE, default)

`alpha` a `numeric` value as the mixing parameter in Elastic Net Regression method (0.5, default)

`show.plot` a `logical` a `logical`. If TRUE, a ROC curve is plotted. Default is TRUE

`direction` a `character` string determines in which direction the comparison will be made. ">": if the predictor values for the control group are higher than the values of the case group (controls > cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < cases).

`conf.level` a `numeric` values determines the confidence interval for the ROC curve(0.95, default).

`cutoff.method` a `character` string determines the cutoff method for the ROC curve.

`...` further arguments. Currently has no effect on the results.

Value

A list of `numeric` nonlinear combination scores calculated according to the given method and standardization option

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
data("exampleData1")
data <- exampleData1

markers <- data[, -1]
status <- factor(data$group, levels = c("not_needed", "needed"))
event <- "needed"
cutoff.method <- "Youden"

score1 <- nonlinComb(
  markers = markers, status = status, event = event,
  method = "lassoreg", include.interact = FALSE, resample = "boot", niters = 5,
  degree1 = 4, degree2 = 4, cutoff.method = cutoff.method,
```

```
  direction = "<"
)

score2 <- nonlinComb(
  markers = markers, status = status, event = event,
  method = "splines", resample = "none", cutoff.method = cutoff.method,
  standardize = "tScore", direction = "<"
)

score3 <- nonlinComb(
  markers = markers, status = status, event = event,
  method = "lassoreg", resample = "repeatedcv", include.interact = TRUE,
  cutoff.method = "ROC01", standardize = "zScore", direction = "auto"
)
```

`plotComb` *Plot the combination scores using the training model*

Description

The `plotComb` a function that generates plots from the training model. The function takes argument model. The outputs of the function are three different plots generated from the combination scores.

Usage

```
plotComb(model, status)
```

Arguments

`model`	a `list` object where the parameters from the training model are saved.
`status`	a `factor` vector that includes the actual disease status of the patients

Value

A `data.frame` plots

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- exampleData1[, -1]
```

```
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))
event <- "needed"

score1 <- linComb(
  markers = markers, status = status, event = event,
  method = "scoring", resample = "none",
  standardize = "none", direction = "<", cutoff.method = "Youden"
)

plotComb(score1, status)

score2 <- nonlinComb(
  markers = markers, status = status, event = event,
  method = "nsgam", resample = "cv", include.interact = FALSE, direction = "<",
  standardize = "zScore", cutoff.method = "Youden"
)

plot.score2 <- plotComb(score2, status)

score3 <- mathComb(
  markers = markers, status = status, event = event,
  method = "distance", distance = "euclidean", direction = "auto",
  standardize = "tScore", cutoff.method = "Youden"
)

plot.score3 <- plotComb(score3, status)
```

`predict.dtComb` *Predict combination scores and labels for new data sets using the training model*

Description

The `predict.dtComb` is a function that generates predictions for a new dataset of biomarkers using the parameters from the fitted model. The function takes arguments newdata and model. The function's output is the combination scores and labels of object type.

Usage

```
## S3 method for class 'dtComb'
predict(object, newdata = NULL, ...)
```

Arguments

`object`	a `list` object where the parameters from the training model are saved.
`newdata`	a `numeric` new data set that includes biomarkers that have not been introduced to the model before.
`...`	further arguments. Currently has no effect on the results.

Value

A data.frame predicted combination scores (or probabilities) and labels

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- exampleData1[, -1]
status <- factor(exampleData1$group, levels = c("not_needed", "needed"))
event <- "needed"

score1 <- linComb(
  markers = markers, status = status, event = event,
  method = "logistic", resample = "none",
  standardize = "none", direction = "<", cutoff.method = "Youden"
)

comb.score1 <- predict(score1, markers)

score2 <- nonlinComb(
  markers = markers, status = status, event = "needed", include.interact = TRUE,
  method = "polyreg", resample = "repeatedcv", nfolds = 5,
  nrepeats = 10, cutoff.method = "Youden", direction = "auto"
)

comb.score2 <- predict(score2, markers)

score3 <- mathComb(
  markers = markers, status = status, event = event,
  method = "distance", distance = "euclidean", direction = "auto",
  standardize = "tScore", cutoff.method = "Youden"
)

comb.score3 <- predict(score3, markers)
```

---

print_train *Print the summary of linComb, nonlinComb, mlComb and mathComb functions.*

Description

The print_train function prints the summary statistics of the fitted model

Usage

```
print_train(print_model)
```

Arguments

print_model	a list of parameters taken from the fitted model that includes the combination method, resampling method, pre-processing method, selected optimum parameters and the results of fit.

Value

No return value writes a summary of the results to the console.

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

---

rocsum — *Generate ROC curves and related statistics for the given markers and Combination score.*

Description

The rocsum function returns the ROC curves with coordinates, Area Under the Curves of markers and combination score, Area Under the Curve comparison of markers and combination score, Confusion matrices for both markers and combination score with the cutoff values derived from the ROC Curves.

Usage

```
rocsum(
  markers = NULL,
  comb.score = NULL,
  status = NULL,
  event = NULL,
  direction = c("auto", "<", ">"),
  conf.level = 0.95,
 cutoff.method = c("CB", "MCT", "MinValueSp", "MinValueSe", "ValueSp", "ValueSe",
  "MinValueSpSe", "MaxSp", "MaxSe", "MaxSpSe", "MaxProdSpSe", "ROC01", "SpEqualSe",
    "Youden", "MaxEfficiency", "Minimax", "MaxDOR", "MaxKappa", "MinValueNPV",
  "MinValuePPV", "ValueNPV", "ValuePPV", "MinValueNPVPPV", "PROC01", "NPVEqualPPV",
    "MaxNPVPPV", "MaxSumNPVPPV", "MaxProdNPVPPV", "ValueDLR.Negative",
  "ValueDLR.Positive", "MinPvalue", "ObservedPrev", "MeanPrev", "PrevalenceMatching"),
  show.plot = show.plot
)
```

Arguments

markers	a numeric data frame that includes two diagnostic tests results
comb.score	a matrix of numeric combination scores calculated according to the given method
status	a factor vector that includes the actual disease status of the patients
event	a character string that indicates the event in the status to be considered as positive event
direction	a character string determines in which direction the comparison will be made. ">": if the predictor values for the control group are higher than the values of the case group (controls > cases). "<": if the predictor values for the control group are lower or equal than the values of the case group (controls < cases).
conf.level	a numeric values determines the confidens interval for the ROC curve(0.95, default).
cutoff.method	a character string determines the cutoff method for the ROC curve.
show.plot	a logical. If TRUE, a ROC curve is plotted. Default is FALSE.

Value

A list of numeric ROC Curves, AUC statistics and Confusion matrices.

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

---

std.test *Standardization according to the training model parameters.*

Description

The std.test Standardization parameters will be taken from the fitted training model and applied to the new data set.

Usage

```
std.test(newdata, model)
```

Arguments

newdata	a numeric data frame of biomarkers
model	a list of parameters from the output of linComb, nonlinComb, mlComb or mathComb functions.

Value

A numeric dataframe of standardized biomarkers

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

---

`std.train` *Standardization according to the chosen method.*

Description

The `std.train` Standardization (range, zScore etc.) can be estimated from the training data and applied to any dataset with the same variables.

Usage

```
std.train(data, standardize = NULL)
```

Arguments

`data`	a `numeric` data frame of biomarkers
`standardize`	a `character` string indicating the name of the standardization method. The default option is no standardization applied. Available options are:

- **Z-score** (`zScore`): This method scales the data to have a mean of 0 and a standard deviation of 1. It subtracts the mean and divides by the standard deviation for each feature. Mathematically,

$$Z - score = \frac{x - (\overline{x})}{sd(x)}$$

where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **T-score** (`tScore`): T-score is commonly used in data analysis to transform raw scores into a standardized form. The standard formula for converting a raw score x into a T-score is:

$$T - score = \left(\frac{x - (\overline{x})}{sd(x)} \times 10\right) + 50$$

where x is the value of a marker, $\overline{x}$ is the mean of the marker and $sd(x)$ is the standard deviation of the marker.

- **Range (a.k.a. min-max scaling)** (`range`): This method transforms data to a specific range, between 0 and 1. The formula for this method is:

$$Range = \frac{x - min(x)}{max(x) - min(x)}$$

- **Mean** (mean): This method, which helps to understand the relative size of a single observation concerning the mean of dataset, calculates the ratio of each data point to the mean value of the dataset.

$$Mean = \frac{x}{\overline{x}}$$

where x is the value of a marker and $\overline{x}$ is the mean of the marker.

- **Deviance** (deviance): This method, which allows for comparison of individual data points in relation to the overall spread of the data, calculates the ratio of each data point to the standard deviation of the dataset.

$$Deviance = \frac{x}{sd(x)}$$

where x is the value of a marker and $sd(x)$ is the standard deviation of the marker.

Value

A numeric data.frame of standardized biomarkers

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
# call data
data(exampleData1)

# define the function parameters
markers <- exampleData1[, -1]
markers2 <- std.train(markers, "deviance")
```

---

transform_math *Mathematical transformations for biomarkers.*

Description

The transform_math function applies a user preference transformation from log exp sin cos transformations for biomarkers.

Usage

```
transform_math(markers, transform)
```

Arguments

markers	a numeric data frame that contains the biomarkers
transform	a numeric string specifying the method used for transform the markers. The available methods are: log exp sin cos.

Value

A numeric dataframe of standardized biomarkers

Author(s)

Serra Ilayda Yerlitas, Serra Bersan Gengec, Necla Kochan, Gozde Erturk Zararsiz, Selcuk Korkmaz, Gokmen Zararsiz

Examples

```
data(exampleData1)
markes <- exampleData1[, -1]
transform_math(markes, transform = "log")
```

Index