Package 'OmicsMarkeR'

October 16, 2019

Title Classification and Feature Selection for 'Omics' Datasets

Description Tools for classification and feature selection for 'omics' level datasets. It is a tool to provide multiple multivariate classification and feature selection techniques complete with multiple stability metrics and aggregation techniques. It is primarily designed for analysis of metabolomics datasets but potentially extendable to proteomics and transcriptomics applications.

Date 2017-01-05

Version 1.16.0

Author Charles E. Determan Jr. <cdetermanjr@gmail.com>

Maintainer Charles E. Determan Jr. <cdetermanjr@gmail.com>

VignetteBuilder knitr

Depends R (>= 3.2.0)

Imports graphics, stats, utils, plyr (>= 1.8), data.table (>= 1.9.4), caret (>= 6.0-37), DiscriMiner (>= 0.1-29), e1071 (>= 1.6-1), randomForest (>= 4.6-10), gbm (>= 2.1), pamr (>= 1.54.1), glmnet (>= 1.9-5), caTools (>= 1.14), foreach (>= 1.4.1), permute (>= 0.7-0), assertive (>= 0.3-0), assertive.base (>= 0.0-1)

Roxygen list(wrap = FALSE)

License GPL-3

LazyData true

URL http://github.com/cdeterman/OmicsMarkeR

BugReports http://github.com/cdeterman/OmicsMarkeR/issues/new

biocViews Metabolomics, Classification, FeatureExtraction

Repository Bioconductor

Suggests testthat, BiocStyle, knitr

RoxygenNote 5.0.1

git_url https://git.bioconductor.org/packages/OmicsMarkeR

git_branch RELEASE_3_9

git_last_commit 8410584

git_last_commit_date 2019-05-02

Date/Publication 2019-10-15

R topics documented:

aggregation *Feature Aggregation*

Description

Compiles matrix of ranked features via user defined 'metric'

Usage

```
aggregation(efs, metric, f = NULL)
```

Arguments

efs	A matrix of selected features
metric	string indicating the type of aggregation. Avialable options are "CLA" (Complete Linear), "EM" (Ensemble Mean), "ES" (Ensemble Stability), and "EE" (Ensemble Exponential)
f	The number of features desired. Default f = NULL

Value

agg	Aggregated list of features

Author(s)

Charles Determan Jr

References

Abeel T., Helleputte T., Van de Peer Y., Dupont P., Saeys Y. (2010) *Robust biomarker identification for cancer diagnosis with ensemble feature selection methods*. Bioinformatics 26(3) 392-398.

Meinshausen N., Buhlmann P. (2010) *Stability selection*. J.R. Statist. Soc. B. 72(4) 417-473.

Haury A., Gestraud P., Vert J. (2011) *The Influence of Features Selection Methods on Accuracy, Stability, and Interpretability of Molecular Signatures*. PLoS ONE 6(12) e28210. doi: 10.1371/journal.pone.0028210.

See Also

CLA, ES, EM, EE

Examples

```
# test data
ranks <- replicate(5, sample(seq(50), 50))
row.names(ranks) <- paste0("V", seq(50))

aggregation(ranks, "CLA")
```

`bagging.wrapper` *Bagging Wrapper for Ensemble Features Selection*

Description

Compiles results of ensemble feature selection

Usage

```
bagging.wrapper(X, Y, method, bags, f, aggregation.metric, k.folds, repeats,
  res, tuning.grid, optimize, optimize.resample, metric, model.features,
  allowParallel, verbose, theDots)
```

Arguments

`X`	A matrix containing numeric values of each feature
`Y`	A factor vector containing group membership of samples
`method`	A vector listing models to be fit
`bags`	Number of bags to be run
`f`	Number of features desired
`aggregation.metric`	string indicating the type of ensemble aggregation. Avialable options are `"CLA"` (Complete Linear), `"EM"` (Ensemble Mean), `"ES"` (Ensemble Stability), and `"EE"` (Ensemble Exponential)
`k.folds`	Number of folds generated during cross-validation
`repeats`	Number of times cross-validation repeated
`res`	Optional - Resolution of model optimization grid
`tuning.grid`	Optional list of grids containing parameters to optimize for each algorithm. Default `"tuning.grid = NULL"` lets function create grid determined by `"res"`
`optimize`	Logical argument determining if each model should be optimized. Default `"optimize = TRUE"`
`optimize.resample`	Logical argument determining if each resample should be re-optimized. Default `"optimize.resample = FALSE"` - Only one optimization run, subsequent models use initially determined parameters
`metric`	Criteria for model optimization. Available options are `"Accuracy"` (Predication Accuracy), `"Kappa"` (Kappa Statistic), and `"AUC-ROC"` (Area Under the Curve - Receiver Operator Curve)
`model.features`	Logical argument if should have number of features selected to be determined by the individual model runs. Default `"model.features = FALSE"`
`allowParallel`	Logical argument dictating if parallel processing is allowed via foreach package. Default `allowParallel = FALSE`
`verbose`	Logical argument if should output progress
`theDots`	Optional arguments provided for specific models or user defined parameters if `"optimize = FALSE"`.

Value

`results` List with the following elements:

- Methods: Vector of models fit to data
- ensemble.results: List of length = length(method) containing aggregated features
- Number.bags: Number of bagging iterations
- Agg.metric: Aggregation method applied
- Number.features: Number of user-defined features

`bestTunes` If `"optimize.resample = TRUE"` then returns list of best parameters for each iteration

Author(s)

Charles Determan Jr

---

canberra *Canberra Distance*

Description

Calculates canberra distance between two vectors. In brief, the higher the canberra distance the greater the 'distance' between the two vectors (i.e. they are less similar).

Usage

```
canberra(x, y)
```

Arguments

x	numeric vector of ranks
y	numeric vector of ranks with compatible length to x

Value

Returns the canberra distance for the two vectors

Note

The `canberra_stability` function is used internally to return the canberra metric.

Author(s)

Charles E. Determan Jr.

References

Jurman G., Merler S., Barla A., Paoli S., Galea A., & Furlanello C. (2008) *Algebraic stability indicators for ranked lists in molecular profiling*. Bioinformatics 24(2): 258-264.

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

Examples

```
# Canberra demo
v1 <- seq(10)
v2 <- sample(v1, 10)
canberra(v1, v2)

canberra_stability(v1, v2)
```

`canberra_stability` *Canberra Stability*

Description

Calculates canberra stability between two ranked lists. In brief, the raw canberra distance is scaled to a [0,1] distribution by the maximum canberra metric. Lastly, this value is subtracted from 1 to provide the same interpretation as the other stability metrics whereby 1 is identical and 0 is no stability.

Usage

```
canberra_stability(x, y)
```

Arguments

x	numeric vector of ranks
y	numeric vector of ranks with compatible length to x

Value

Returns the canberra stability for the two vectors

Author(s)

Charles E. Determan Jr.

References

Jurman G., Merler S., Barla A., Paoli S., Galea A., & Furlanello C. (2008) *Algebraic stability indicators for ranked lists in molecular profiling*. Bioinformatics 24(2): 258-264.

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

Examples

```
# Canberra demo
v1 <- seq(10)
v2 <- sample(v1, 10)
canberra(v1, v2)

canberra_stability(v1, v2)
```

CLA *Complete Linear Aggregation*

Description

Compiles matrix of ranked features via complete linear aggregation

Usage

```
CLA(efs, f)
```

Arguments

efs	A matrix of selected features
f	The number of features desired. If rank correlation desired, f = NULL

Value

agg	Aggregated list of features

Author(s)

Charles Determan Jr

References

Abeel T., Helleputte T., Van de Peer Y., Dupont P., Saeys Y. (2010) *Robust biomarker identification for cancer diagnosis with ensemble feature selection methods*. Bioinformatics 26:3 392-398.

See Also

ES, EM, EE

create.corr.matrix *Correlated Multivariate Data Generator*

Description

Generates a matrix of dimensions dim(U) with induced correlations. Blocks of variables are randomly assigned and correlations are induced. A noise matrix is applied to the final matrix to perturb 'perfect' correlations.

Usage

```
create.corr.matrix(U, k = 4, min.block.size = 2, max.block.size = 5)
```

Arguments

U	Numeric matrix
k	Correlation Perturbation - The higher k, the more the data is perturbed. Default k = 4
min.block.size	minimum number of variables to correlate Default min.block.size = 2
max.block.size	maximum number of variables to correlate Default max.block.size = 5

Value

A numberic matrix of dimension dim(U) with correlations induced between variables

Note

Output does not contain classes, may provide externally as classes are irrelevant in this function.

Author(s)

Charles E. Determan Jr.

References

Wongravee, K., Lloyd, G R., Hall, J., Holmboe, M. E., & Schaefer, M. L. (2009). *Monte-Carlo methods for determining optimal number of significant variables. Application to mouse urinary profiles.* Metabolomics, 5(4), 387-406. http://dx.doi.org/10.1007/s11306-009-0164-4

See Also

create.random.matrix, create.discr.matrix

Examples

```
# Create Multivariate Matrices

# Random Multivariate Matrix

# 50 variables, 100 samples, 1 standard devation, 0.2 noise factor

rand.mat <- create.random.matrix(nvar = 50,
                                 nsamp = 100,
                                 st.dev = 1,
                                 perturb = 0.2)


# Induce correlations in a numeric matrix

# Default settings
# minimum and maximum block sizes (min.block.size = 2, max.block.size = 5)
# default correlation purturbation (k=4)
# see ?create.corr.matrix for citation for methods

corr.mat <- create.corr.matrix(rand.mat)


# Induce Discriminatory Variables
```

```
# 10 discriminatory variables (D = 10)
# default discrimination level (l = 1.5)
# default number of groups (num.groups=2)
# default correlation purturbation (k = 4)

dat.discr <- create.discr.matrix(corr.mat, D=10)
```

`create.discr.matrix` *Discriminatory Multivariate Data Generator*

Description

Generates a matrix of dimensions `dim(U)` with induced correlations. `D` variables are randomly selected as discriminatory. If `num.groups = 2` then discrimination is induced by adding and subtracting values derived from the level of of discrimination, `l`, for the classes respectively. Multi-class datasets have a few further levels of randomization. For each variable, a random number of the groups are selected as discriminating while the remaining groups are not altered. For each discriminatory group, a unique change is provided by randomly assigning addition or subtraction of the discrimination factor. For example, if 3 groups are selected and two groups are assigned as addition and the third subtraction, the second addition is multiplied by its number of replicates. E.g. (1,1,-1) -> (1,2,-1). These values are randomized and then multiplied by the respective discrimination factor. The resulting values are then added/subtracted from the respective groups. A noise matrix is applied to the final matrix to perturb 'perfect' discrimination.

Usage

```
create.discr.matrix(V, D = 20, l = 1.5, num.groups = 2, k = 4)
```

Arguments

`V`	Numeric matrix
`D`	Number of discriminatory variables induced. Default `D = 20`
`l`	Level of discrimination, higher = greater separation. Default `l = 1.5`
`num.groups`	Number of groups in the dataset
`k`	Correlation Perturbation - The higher k, the more the data is perturbed. Default `k = 4`

Value

List of the following elements

`discr.mat`	Matrix of dimension `dim(V)+1` with discriminatory variables induced and the .classes added to the end of the matrix.
`features`	Vector of features that were induced to be discriminatory.

Author(s)

Charles E. Determan Jr.

References

Wongravee, K., Lloyd, G R., Hall, J., Holmboe, M. E., & Schaefer, M. L. (2009). *Monte-Carlo methods for determining optimal number of significant variables. Application to mouse urinary profiles.* Metabolomics, 5(4), 387-406. http://dx.doi.org/10.1007/s11306-009-0164-4

Examples

```
# Create Multivariate Matrices

# Random Multivariate Matrix

# 50 variables, 100 samples, 1 standard devation, 0.2 noise factor

rand.mat <- create.random.matrix(nvar = 50,
                                 nsamp = 100,
                                 st.dev = 1,
                                 perturb = 0.2)


# Induce correlations in a numeric matrix

# Default settings
# minimum and maximum block sizes (min.block.size = 2, max.block.size = 5)
# default correlation purturbation (k=4)
# see ?create.corr.matrix for citation for methods

corr.mat <- create.corr.matrix(rand.mat)


# Induce Discriminatory Variables

# 10 discriminatory variables (D = 10)
# default discrimination level (l = 1.5)
# default number of groups (num.groups=2)
# default correlation purturbation (k = 4)

dat.discr <- create.discr.matrix(corr.mat, D=10)
```

`create.random.matrix` *Random Multivariate Data Generator*

Description

Generates a matrix of dimensions `nvar` by `nsamp` consisting of random numbers generated from a normal distriubtion. This normal distribution is then perturbed to more accurately reflect experimentally acquired multivariate data.

Usage

```
create.random.matrix(nvar, nsamp, st.dev = 1, perturb = 0.2)
```

Arguments

nvar	Number of features (i.e. variables)
nsamp	Number of samples
st.dev	The variation (i.e. standard deviation) that is typical in datasets of interest to the user. Default spread = 1
perturb	The amount of perturbation to the normal distribution. Default perturb = 0.2

Value

Matrix of dimension nvar by nsamp

Author(s)

Charles E. Determan Jr.

References

Wongravee, K., Lloyd, G R., Hall, J., Holmboe, M. E., & Schaefer, M. L. (2009). *Monte-Carlo methods for determining optimal number of significant variables. Application to mouse urinary profiles.* Metabolomics, 5(4), 387-406. http://dx.doi.org/10.1007/s11306-009-0164-4

See Also

create.corr.matrix, create.discr.matrix

Examples

```
# Create Multivariate Matrices

# Random Multivariate Matrix

# 50 variables, 100 samples, 1 standard devation, 0.2 noise factor

rand.mat <- create.random.matrix(nvar = 50,
                                 nsamp = 100,
                                 st.dev = 1,
                                 perturb = 0.2)


# Induce correlations in a numeric matrix

# Default settings
# minimum and maximum block sizes (min.block.size = 2, max.block.size = 5)
# default correlation purturbation (k=4)
# see ?create.corr.matrix for citation for methods

corr.mat <- create.corr.matrix(rand.mat)


# Induce Discriminatory Variables

# 10 discriminatory variables (D = 10)
# default discrimination level (l = 1.5)
# default number of groups (num.groups=2)
```

```
# default correlation purturbation (k = 4)

dat.discr <- create.discr.matrix(corr.mat, D=10)
```

---

denovo.grid *Denovo Grid Generation*

Description

Greates grid for optimizing selected models

Usage

```
denovo.grid(data, method, res)
```

Arguments

data	data of method to be tuned
method	vector indicating the models to generate grids. Available options are "plsda" (Partial Least Squares Discriminant Analysis), "rf" (Random Forest), "gbm" (Gradient Boosting Machine), "svm" (Support Vector Machines), "glmnet" (Elastic-net Generalized Linear Model), and "pam" (Prediction Analysis of Microarrays)
res	Resolution of model optimization grid.

Value

A list containing dataframes of all combinations of parameters for each model:

Author(s)

Charles Determan Jr

See Also

"expand.grid" for generating grids of specific parameters desired. However, NOTE that you must still convert the generated grid to a list.

Examples

```
# random test data
set.seed(123)
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

df <- data.frame(dat.discr$discr.mat, .classes = dat.discr$classes)
```

```
# create tuning grid
denovo.grid(df, "gbm", 3)
```

EE *Ensemble Exponential Aggregation*

Description

Compiles matrix of ranked features via ensemble exponential aggregation

Usage

```
EE(efs, f)
```

Arguments

efs	A matrix of selected features
f	The number of features desired. If rank correlation desired, f = NULL

Value

agg	Aggregated list of features

Author(s)

Charles Determan Jr

References

Haury A., Gestraud P., Vert J. (2011) *The Influence of Features Selection Methods on Accuracy, Stability, and Interpretability of Molecular Signatures*. PLoS ONE 6(12) e28210. doi: 10.1371/journal.pone.0028210

See Also

CLA, ES, EM,

EM *Ensemble Mean Aggregation*

Description

Compiles matrix of ranked features via ensemble mean aggregation

Usage

```
EM(efs, f)
```

Arguments

efs	A matrix of selected features
f	The number of features desired. If rank correlation desired, f = NULL

Value

agg	Aggregated list of features

Author(s)

Charles Determan Jr

References

Abeel T., Helleputte T., Van de Peer Y., Dupont P., Saeys Y. (2010) *Robust biomarker identification for cancer diagnosis with ensemble feature selection methods*. Bioinformatics 26:3 392-398.

See Also

CLA, ES, EE

ES *Ensemble Stability Aggregation*

Description

Compiles matrix of ranked features via ensemble stability aggregation

Usage

```
ES(efs, f)
```

Arguments

efs	A matrix of selected features
f	The number of features desired. If rank correlation desired, f = NULL

Value

agg	Aggregated list of features

Author(s)

Charles Determan Jr

References

Meinshausen N., Buhlmann P. (2010) *Stability selection*. J.R. Statist. Soc. B. 72:4 417-473.

See Also

CLA, EM, EE

extract.args *Argument extractor*

Description

Extract arguments from previously fs.stability models

Usage

```
extract.args(fs.model, method)
```

Arguments

fs.model	Previously fit fs.stability model
method	Which model to extract from

Value

args List of model arguments

extract.features *Feature Extraction*

Description

Extracts features from models that have been previously fit.

Usage

```
extract.features(x, dat = NULL, grp = NULL, method,
  model.features = FALSE, bestTune = NULL, f, comp.catch = NULL)
```

Arguments

x	Previously fitted model
dat	Numeric variable data used for fitted models (In appropriate format)
grp	Vector of training classes
method	String indicating the INDIVIDUAL model being extracted from
model.features	Logical argument dictating if features selected determined by models instead of user determined number of features.
bestTune	If model.features = TRUE, must provide the parameter at which to extract features from the model.
f	Number of features to subset
comp.catch	An internal check for plsda models. If the optimal model contains only 1 component, the ncomp paramter must be set to 2 for the model. However, features are still extracted only from the first component.

Value

Returns list of the features selected from the fitted model.

`feature.table` *Feature Consistency Table*

Description

Extracts and sorts the features identified for a given method.

Usage

```
feature.table(features, method)
```

Arguments

`features`	A fs.stability fitted object
`method`	Algorithm of interest Available options are `"plsda"` (Partial Least Squares Discriminant Analysis), `"rf"` (Random Forest), `"gbm"` (Gradient Boosting Machine), `"svm"` (Support Vector Machines), `"glmnet"` (Elastic-net Generalized Linear Model), and `"pam"` (Prediction Analysis of Microarrays)

Value

A data frame containing:

`features`	Features identified by model
`consistency`	Number of iterations feature was identified
`frequency`	Frequency of iterations the feature was identified

Author(s)

Charles Determan Jr

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')

feature.table(fits, "plsda")
```

`fit.only.model` *Fit Models without Feature Selection*

Description

Applies models to high-dimensional data for classification.

Usage

```
fit.only.model(X, Y, method, p = 0.9, optimize = TRUE, tuning.grid = NULL,
  k.folds = if (optimize) 10 else NULL, repeats = if (optimize) 3 else NULL,
  resolution = if (optimize) 3 else NULL, metric = "Accuracy",
  allowParallel = FALSE, verbose = "none", ...)
```

Arguments

`X`	A scaled matrix or dataframe containing numeric values of each feature
`Y`	A factor vector containing group membership of samples
`method`	A vector listing models to be fit. Available options are `"plsda"` (Partial Least Squares Discriminant Analysis), `"rf"` (Random Forest), `"gbm"` (Gradient Boosting Machine), `"svm"` (Support Vector Machines), `"glmnet"` (Elastic-net Generalized Linear Model), and `"pam"` (Prediction Analysis of Microarrays)
`p`	Percent of data to by 'trained'
`optimize`	Logical argument determining if each model should be optimized. Default `"optimize = TRUE"`
`tuning.grid`	Optional list of grids containing parameters to optimize for each algorithm. Default `"tuning.grid = NULL"` lets function create grid determined by `"res"`
`k.folds`	Number of folds generated during cross-validation. Default `"k.folds = 10"`
`repeats`	Number of times cross-validation repeated. Default `"repeats = 3"`
`resolution`	Resolution of model optimization grid. Default `"resolution = 3"`
`metric`	Criteria for model optimization. Available options are `"Accuracy"` (Predication Accuracy), `"Kappa"` (Kappa Statistic), and `"AUC-ROC"` (Area Under the Curve - Receiver Operator Curve)
`allowParallel`	Logical argument dictating if parallel processing is allowed via foreach package. Default `allowParallel = FALSE`
`verbose`	Logical argument if should output progress
`...`	Extra arguments that the user would like to apply to the models

Value

`Methods`	Vector of models fit to data
`performance`	Performance metrics of each model and bootstrap iteration
`specs`	List with the following elements:

- total.samples: Number of samples in original dataset
- number.features: Number of features in orginal dataset
- number.groups: Number of groups
- group.levels: The specific levels of the groups
- number.observations.group: Number of observations in each group

Author(s)

Charles Determan Jr

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fit <- fit.only.model(X=vars,
                      Y=groups,
                      method="plsda",
                      p = 0.9)
```

---

fs.ensembl.stability *Ensemble Classification & Feature Selection*

Description

Applies ensembles of models to high-dimensional data to both classify and determine important features for classification. The function bootstraps a user-specified number of times to facilitate stability metrics of features selected thereby providing an important metric for biomarker investigations, namely whether the important variables can be identified if the models are refit on 'different' data.

Usage

```
fs.ensembl.stability(X, Y, method, k = 10, p = 0.9,
  f = ceiling(ncol(X)/10), bags = 40, aggregation.metric = "CLA",
  stability.metric = "jaccard", optimize = TRUE,
  optimize.resample = FALSE, tuning.grid = NULL, k.folds = if (optimize)
  10 else NULL, repeats = if (k.folds == "LOO") NULL else if (optimize) 3 else
  NULL, resolution = if (optimize) 3 else NULL, metric = "Accuracy",
  model.features = FALSE, allowParallel = FALSE, verbose = "none", ...)
```

Arguments

X	A matrix containing numeric values of each feature
Y	A factor vector containing group membership of samples
method	A vector listing models to be fit. Available options are "plsda" (Partial Least Squares Discriminant Analysis), "rf" (Random Forest), "gbm" (Gradient Boosting Machine), "svm" (Support Vector Machines), "glmnet" (Elastic-net Generalized Linear Model), and "pam" (Prediction Analysis of Microarrays)

k	Number of bootstrapped interations
p	Percent of data to by 'trained'
f	Number of features desired. Default is top 10 "f = ceiling(ncol(variables)/10)". If rank correlation is desired, set "f = NULL"
bags	Number of iterations for ensemble bagging. Default "bags = 40"
aggregation.metric	String indicating which aggregation metric for features selected during bagging. Avialable options are "CLA" (Complete Linear), "EM" (Ensemble Mean), "ES" (Ensemble Stability), and "EE" (Ensemble Exponential)
stability.metric	string indicating the type of stability metric. Avialable options are "jaccard" (Jaccard Index/Tanimoto Distance), "sorensen" (Dice-Sorensen's Index), "ochiai" (Ochiai's Index), "pof" (Percent of Overlapping Features), "kuncheva" (Kuncheva's Stability Measures), "spearman" (Spearman Rank Correlation), and "canberra" (Canberra Distance)
optimize	Logical argument determining if each model should be optimized. Default "optimize = TRUE"
optimize.resample	Logical argument determining if each resample should be re-optimized. Default "optimize.resample = FALSE" - Only one optimization run, subsequent models use initially determined parameters
tuning.grid	Optional list of grids containing parameters to optimize for each algorithm. Default "tuning.grid = NULL" lets function create grid determined by "res"
k.folds	Number of folds generated during cross-validation. May optionally be set to "LOO" for leave-one-out cross-validation. Default "k.folds = 10"
repeats	Number of times cross-validation repeated. Default "repeats = 3"
resolution	Optional - Resolution of model optimization grid. Default "res = 3"
metric	Criteria for model optimization. Available options are "Accuracy" (Predication Accuracy), "Kappa" (Kappa Statistic), and "AUC-ROC" (Area Under the Curve - Receiver Operator Curve)
model.features	Logical argument if should have number of features selected to be determined by the individual model runs. Default "model.features = FALSE"
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package. Default allowParallel = FALSE
verbose	Character argument specifying how much output progress to print. Options are 'none', 'minimal' or 'full'.
...	Extra arguments that the user would like to apply to the models

Value

Methods	Vector of models fit to data
performance	Performance metrics of each model and bootstrap iteration
RPT	Robustness-Performance Trade-Off as defined in Saeys 2008
features	List concerning features determined via each algorithms feature selection criteria.

- metric: Stability metric applied

- features: Matrix of selected features
- stability: Matrix of pairwise comparions and average stability

`stability.models`
: Function perturbation metric - i.e. how similar are the features selected by each model.

`all.tunes`
: If "`optimize.resample = TRUE`" then returns list of optimized parameters for each bagging and bootstrap interation.

`final.best.tunes`
: If "`optimize.resample = TRUE`" then returns list of optimized parameters for each bootstrap of the bagged models refit to aggregated selected features.

`specs`
: List with the following elements:

- total.samples: Number of samples in original dataset
- number.features: Number of features in orginal dataset
- number.groups: Number of groups
- group.levels: The specific levels of the groups
- number.observations.group: Number of observations in each group

Author(s)

Charles Determan Jr

References

Saeys Y., Abeel T., et. al. (2008) *Machine Learning and Knowledge Discovery in Databases*. 313-325. http://link.springer.com/chapter/10.1007/978-3-540-87481-2_21

Examples

```
## Not run:
fits <- fs.ensembl.stability(vars,
groups,
method = c("plsda", "rf"),
f = 10,
k = 3,
k.folds = 10,
verbose = 'none')

## End(Not run)
```

`fs.stability` *Classification & Feature Selection*

Description

Applies models to high-dimensional data to both classify and determine important features for classification. The function bootstraps a user-specified number of times to facilitate stability metrics of features selected thereby providing an important metric for biomarker investigations, namely whether the important variables can be identified if the models are refit on 'different' data.

Usage

```
fs.stability(X, Y, method, k = 10, p = 0.9, f = NULL,
  stability.metric = "jaccard", optimize = TRUE,
  optimize.resample = FALSE, tuning.grid = NULL, k.folds = if (optimize)
  10 else NULL, repeats = if (k.folds == "LOO") NULL else if (optimize) 3 else
  NULL, resolution = if (is.null(tuning.grid) && optimize) 3 else NULL,
  metric = "Accuracy", model.features = FALSE, allowParallel = FALSE,
  verbose = "none", ...)
```

Arguments

X	A scaled matrix or dataframe containing numeric values of each feature
Y	A factor vector containing group membership of samples
method	A vector listing models to be fit. Available options are "plsda" (Partial Least Squares Discriminant Analysis), "rf" (Random Forest), "gbm" (Gradient Boosting Machine), "svm" (Support Vector Machines), "glmnet" (Elastic-net Generalized Linear Model), and "pam" (Prediction Analysis of Microarrays)
k	Number of bootstrapped interations
p	Percent of data to by 'trained'
f	Number of features desired. If rank correlation is desired, set "f = NULL"
stability.metric	string indicating the type of stability metric. Avialable options are "jaccard" (Jaccard Index/Tanimoto Distance), "sorensen" (Dice-Sorensen's Index), "ochiai" (Ochiai's Index), "pof" (Percent of Overlapping Features), "kuncheva" (Kuncheva's Stability Measures), "spearman" (Spearman Rank Correlation), and "canberra" (Canberra Distance)
optimize	Logical argument determining if each model should be optimized. Default "optimize = TRUE"
optimize.resample	Logical argument determining if each resample should be re-optimized. Default "optimize.resample = FALSE" - Only one optimization run, subsequent models use initially determined parameters
tuning.grid	Optional list of grids containing parameters to optimize for each algorithm. Default "tuning.grid = NULL" lets function create grid determined by "res"
k.folds	Number of folds generated during cross-validation. May optionally be set to "LOO" for leave-one-out cross-validation. Default "k.folds = 10"
repeats	Number of times cross-validation repeated. Default "repeats = 3"
resolution	Resolution of model optimization grid. Default "resolution = 3"
metric	Criteria for model optimization. Available options are "Accuracy" (Predication Accuracy), "Kappa" (Kappa Statistic), and "AUC-ROC" (Area Under the Curve - Receiver Operator Curve)
model.features	Logical argument if should have number of features selected to be determined by the individual model runs. Default "model.features = FALSE"
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package. Default allowParallel = FALSE
verbose	Character argument specifying how much output progress to print. Options are 'none', 'minimal' or 'full'.
...	Extra arguments that the user would like to apply to the models

Value

`Methods` Vector of models fit to data

`performance` Performance metrics of each model and bootstrap iteration

`RPT` Robustness-Performance Trade-Off as defined in Saeys 2008

`features` List concerning features determined via each algorithms feature selection criteria.

- metric: Stability metric applied
- features: Matrix of selected features
- stability: Matrix of pairwise comparions and average stability

`stability.models`
Function perturbation metric - i.e. how similar are the features selected by each model.

`original.best.tunes`
If "`optimize.resample = TRUE`" then returns list of optimized parameters for each bootstrap.

`final.best.tunes`
If "`optimize.resample = TRUE`" then returns list of optimized parameters for each bootstrap of models refit to selected features.

`specs` List with the following elements:

- total.samples: Number of samples in original dataset
- number.features: Number of features in orginal dataset
- number.groups: Number of groups
- group.levels: The specific levels of the groups
- number.observations.group: Number of observations in each group

Author(s)

Charles Determan Jr

References

Saeys Y., Abeel T., et. al. (2008) *Machine Learning and Knowledge Discovery in Databases*. 313-325. http://link.springer.com/chapter/10.1007/978-3-540-87481-2_21

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes
```

```
fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')
```

`jaccard` *Jaccard Index*

Description

Calculates jaccard index between two vectors of features. In brief, the closer to 1 the more similar the vectors. The two vectors may have an arbitrary cardinality (i.e. don't need same length). Also known as the Tanimoto distance metric. Defined as the size of the vectors' intersection divided by the size of the union of the vectors.

Usage

```
jaccard(x, y)
```

Arguments

x	vector of feature names
y	vector of feature names

Value

Returns the jaccard index for the two vectors. It takes values in [0,1], with 0 meaning no overlap between two sets and 1 meaning two sets are identical.

Author(s)

Charles E. Determan Jr.

References

Jaccard P. (1908) *Nouvelles recherches sur la distribution florale*. Bull. Soc. Vaudoise Sci. Nat. 44: 223-270.

Real R. & Vargas J.M. (1996) *The Probabilistic Basis of Jaccard's Index of Similarity* Systematic Biology 45(3): 380-385.

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

`kuncheva`, `sorensen`, `ochiai`, `pof`, `pairwise.stability`, `pairwise.model.stability`

Examples

```
# Jaccard demo
v1 <- paste("Metabolite", seq(10), sep="_")
v2 <- sample(v1, 10)
jaccard(v1, v2)
```

---

kuncheva *Kuncheva's Index*

Description

Calculates Kuncheva's index between two vectors of features. In brief, the closer to 1 the more similar the vectors. The two vectors must have the same cardinality (i.e. same length).

Usage

```
kuncheva(x, y, num.features)
```

Arguments

x	Character vector of feature names
y	Character vector of feature names
num.features	total number of features in the original dataset

Value

Returns the Kuncheva Index for the two vectors. It takes values in [0,1], with 0 meaning no overlap between two sets and 1 meaning two sets are identical.

Note

The returned Kuncheva Index has been scaled from its original [-1,1] range to [0,1] in order to make it compatible with RPT.

Author(s)

Charles E. Determan Jr.

References

Kuncheva L. (2007) *A stability index for feature selection*. Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications. pp. 390-395.

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

kuncheva, sorensen, ochiai, pof, pairwise.stability, pairwise.model.stability

Examples

```
# Kuncheva demo
# Assuming 50 metabolites were measured
# But only 10 were found significant

# For demonstration purposes only!!!
some.numbers <- seq(20)

# Metabolites identified from one run
v1 <- paste("Metabolite", sample(some.numbers, 10), sep="_")
# Metabolites identifed from second run
v2 <- paste("Metabolite", sample(some.numbers, 10), sep="_")
kuncheva(v1, v2, 50)
```

---

modelList *Model List*

Description

Provide a list of currently implemented methods for OmicsMarkeR.

Usage

```
modelList()
```

Value

A data.frame containing:

`methods`	The abbreviated code for the method
`description`	Full name of the method

Author(s)

Charles Determan Jr.

Examples

```
modelList()
```

modelTuner *Model Tuner*

Description

Optimizes each model based upon the parameters provided either by the internal denovo.grid function or by the user.

Usage

```
modelTuner(trainData, guide, method, inTrain, outTrain, lev,
  savePredictions = FALSE, allowParallel = FALSE, verbose = "none",
  theDots = NULL)
```

Arguments

trainData	Data used to fit the model
guide	Output from tune.instructions. Facilitates the optimization by avoiding redundant model fitting.
method	Vector of strins listing models to be fit
inTrain	Indicies for cross-validated training models
outTrain	Indicies for cross-validated testing models
lev	Group levels
savePredictions	Logical argument dictating if should save the prediction data. Default savePredictions = FALSE
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package
verbose	Character argument specifying how much output progress to print. Options are 'none', 'minimal' or 'full'.
theDots	List of additional arguments provided in the initial classification and features selection function

Value

Returns list of fitted models

Author(s)

Charles E. Determan Jr.

modelTuner_loo *Model Tuner for Leave-One-Out Cross-Validation*

Description

Optimizes each model via LOO CV based upon the parameters provided either by the internal denovo.grid function or by the user.

Usage

```
modelTuner_loo(trainData, guide, method, inTrain, outTrain, lev,
  savePredictions = FALSE, allowParallel = FALSE, verbose = "none",
  theDots = NULL)
```

Arguments

trainData	Data used to fit the model
guide	Output from tune.instructions. Facilitates the optimization by avoiding redundant model fitting.
method	Vector of strins listing models to be fit
inTrain	Indicies for cross-validated training models
outTrain	Indicies for cross-validated testing models
lev	Group levels
savePredictions	Logical argument dictating if should save the prediction data. Default savePredictions = FALSE
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package
verbose	Character argument specifying how much output progress to print. Options are 'none', 'minimal' or 'full'.
theDots	List of additional arguments provided in the initial classification and features selection function

Value

Returns list of fitted models

Author(s)

Charles E. Determan Jr.

noise.matrix *Noise Matrix Generator*

Description

Provides a matrix to perturb randomly generated data to facilitate a more realistic dataset.

Usage

```
noise.matrix(matrix, k)
```

Arguments

matrix	A matrix of simulated data with dimensions comparable to 'real' datasets
k	Correlation Perturbation - The higher k, the more the data is perturbed.

Value

Returns a matrix of the same dimensions as `matrix` that can add to perturb the original simulated data.

Author(s)

Charles E. Determan Jr.

ochiai *Ochiai's Index*

Description

Calculates Ochiai's index between two vectors of features. In brief, the closer to 1 the more similar the vectors. The two vectors may have an arbitrary cardinality (i.e. don't need same length). Very similar to the Jaccard Index `jaccard` but Ochiai is a geometric means of the ratio.

Usage

```
ochiai(x, y)
```

Arguments

x	Character vector of feature names
y	Character vector of feature names

Value

Returns the Ochiai Index for the two vectors. It takes values in [0,1], with 0 meaning no overlap between two sets and 1 meaning two sets are identical.

Author(s)

Charles E. Determan Jr.

References

Ochiai A. (1957) *Zoogeographical studies on the soleoid fishes found in Japan and its neigbouring regions*. Bulletin of the Japanese Society of Scientific Fisheries. 22: 526-530.

Zucknick M., Richardson S., & Stronach E.A. (2008) *Comparing the characteristics of gene expression profiles derived by univariate and multivariate classification methods*. Statistical Applications in Genetics and Molecular Biology. 7(1): Article 7. doi:10.2202/1544-6115.1307

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

kuncheva, sorensen, ochiai, pof, pairwise.stability, pairwise.model.stability

Examples

```
# Ochiai demo
v1 <- paste("Metabolite", seq(10), sep="_")
v2 <- sample(v1, 10)
ochiai(v1, v2)
```

---

optimize.model *Model Optimization and Metrics*

Description

Optimizes each model based upon the parameters provided either by the internal denovo.grid function or by the user.

Usage

```
optimize.model(trainVars, trainGroup, method, k.folds = 10, repeats = 3,
  res = 3, grid = NULL, metric = "Accuracy", allowParallel = FALSE,
  verbose = "none", theDots = NULL)
```

Arguments

trainVars	Data used to fit the model
trainGroup	Group identifiers for the training data
method	A vector of strings listing models to be optimized
k.folds	Number of folds generated during cross-validation. Default "k.folds = 10"
repeats	Number of times cross-validation repeated. Default "repeats = 3"
res	Resolution of model optimization grid. Default "res = 3"
grid	Optional list of grids containing parameters to optimize for each algorithm. Default "grid = NULL" lets function create grid determined by "res"
metric	Criteria for model optimization. Available options are "Accuracy" (Predication Accuracy), "Kappa" (Kappa Statistic), and "AUC-ROC" (Area Under the Curve - Receiver Operator Curve)
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package

verbose	Character argument specifying how much output progress to print. Options are 'none', 'minimal' or 'full'.
theDots	List of additional arguments provided in the initial classification and features selection function

Value

Basically a list with the following elements:

method	Vector of strings listing models that were optimized
performance	Performance generated internally to optimize model
bestTune	List of paramaters chosen for each model
dots	List of extra arguments initially provided
metric	Criteria that was used for model optimization
finalModels	The fitted models with the 'optimum' parameters
performance.metrics	The performance metrics calculated internally for each resulting prediction
tune.metrics	The results from each tune
perfNames	The names of the performance metrics
comp.catch	If the optimal PLSDA model contains only 1 component, the model must be refit with 2 components. This catches the 1 component parameter so feature selection and further performance analysis can be conducted on the 1 component.

Author(s)

Charles E. Determan Jr.

---

pairwise.model.stability

Pairwise Model Stability Metrics

Description

Conducts all pairwise comparisons of each model's selected features selected following bootstrapping. Also known as the function perturbation ensemble approach

Usage

```
pairwise.model.stability(features, stability.metric, nc)
```

Arguments

features	A matrix of selected features
stability.metric	string indicating the type of stability metric. Avialable options are "jaccard" (Jaccard Index/Tanimoto Distance), "sorensen" (Dice-Sorensen's Index), "ochiai" (Ochiai's Index), "pof" (Percent of Overlapping Features), "kuncheva" (Kuncheva's Stability Measures), "spearman" (Spearman Rank Correlation), and "canberra" (Canberra Distance)
nc	Number of original features

Value

A list is returned containing:

comparisons	Matrix of pairwise comparisons
overall	The average of all pairwise comparisons

Author(s)

Charles Determan Jr

References

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

pairwise.stability

Examples

```
# pairwise.model.stability demo
# For demonstration purposes only!!!
some.numbers <- seq(20)

# A list containing the metabolite matrices for each algorithm
# As an example, let's say we have the output from two different models
# such as plsda and random forest.
# matrix of Metabolites identified (e.g. 5 trials)
plsda <-
    replicate(5, paste("Metabolite", sample(some.numbers, 10), sep="_"))
rf <-
    replicate(5, paste("Metabolite", sample(some.numbers, 10), sep="_"))

features <- list(plsda=plsda, rf=rf)

# nc may be omitted unless using kuncheva
pairwise.model.stability(features, "kuncheva", nc=20)
```

---

pairwise.stability *Pairwise Stability Metrics*

Description

Conducts all pairwise comparisons of features selected following bootstrapping. Also known as the data perturbation ensemble approach.

Usage

```
pairwise.stability(features, stability.metric, nc)
```

Arguments

features	A matrix of selected features
stability.metric	string indicating the type of stability metric.
nc	Optional argument to be used with 'kuncheva' stability. Refers to the number of variables in original data. Available options are "jaccard" (Jaccard Index/Tanimoto Distance), "sorensen" (Dice-Sorensen's Index), "ochiai" (Ochiai's Index), "pof" (Percent of Overlapping Features), "kuncheva" (Kuncheva's Stability Measures), "spearman" (Spearman Rank Correlation), and "canberra" (Canberra Distance) @param nc Number of variables in original dataset

Value

A list is returned containing:

comparisons	Matrix of pairwise comparisons
overall	The average of all pairwise comparisons

Author(s)

Charles Determan Jr

References

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

Examples

```
# pairwise.stability demo

# For demonstration purposes only!!!
some.numbers <- seq(20)

# matrix of Metabolites identified (e.g. 5 trials)
features <-
    replicate(5, paste("Metabolite", sample(some.numbers, 10), sep="_"))

# nc may be omitted unless using kuncheva
pairwise.stability(features, "jaccard")
```

---

params *Model Parameters and Properties*

Description

Provides a list of the models with their respective parameters and properties.

Usage

```
params(method = NULL)
```

Arguments

method	A vector of strings listing the models to be returned

Value

Returns a dataframe of the following components:

method A vector of strings listing models returned

parameter A vector of possible parameters to be optimized

label A vector of the names for each possible parameter

seq A logical indicator if the parameter is sequential in the model (i.e. if model is able to fit all 'lower' parameters simultaneously)

Examples

```
params("plsda")
```

---

perf.calc *Performance Statistics Calculations*

Description

Calculates confusion matrix and ROC statistics comparing the results of the fitted models to the observed groups.

Usage

```
perf.calc(data, lev = NULL, model = NULL)
```

Arguments

data	dataframe of predicted (pred) and observed (obs) groups
lev	Group levels
model	String indicating which model was initially run

Value

Returns confusion matrix and ROC performance statistics including Accuracy, Kappa, ROC.AUC, Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value

See Also

caret function confusionMatrix

`performance.metrics` *Performance Metrics of fs.stability or fs.ensembl.stability object*

Description

This will provide a concise data.frame of confusion matrix and ROC statistics from the results of `fs.stability` or `fs.ensembl.stability`.

Usage

```
performance.metrics(fit.model, digits = max(3, getOption("digits") - 3))
```

Arguments

`fit.model`	An fs.stability or fs.ensembl.stability object
`digits`	How many digits to round values

Value

Dataframe of performance statistics by model

Author(s)

Charles E. Determan Jr.

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')

performance.metrics(fits)
```

performance.stats *Performance Statistics (Internal for* perf.calc*)*

Description

Calculates confusion matrix and ROC statistics comparing the results of the fitted models to the observed groups.

Usage

```
performance.stats(pred, obs)
```

Arguments

pred	vector of groups predicted by a fitted classification model
obs	vector of groups from the original dataset

Value

Returns confusion matrix and ROC performance statistics including Accuracy, Kappa, ROC.AUC, Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value

See Also

caret function confusionMatrix

perm.class *Monte Carlo Permutation of Model Performance*

Description

Applies Monte Carlo permutations to user specified models. The user can either use the results from fs.stability or provide specified model parameters.

Usage

```
perm.class(fs.model = NULL, X, Y, method, k.folds = 5,
  metric = "Accuracy", nperm = 10, allowParallel = FALSE,
  create.plot = FALSE, verbose = TRUE, ...)
```

Arguments

fs.model	Object containing results from fs.stability
X	A scaled matrix or dataframe containing numeric values of each feature
Y	A factor vector containing group membership of samples
method	A string of the model to be fit. Available options are "plsda" (Partial Least Squares Discriminant Analysis), "rf" (Random Forest), "gbm" (Gradient Boosting Machine), "svm" (Support Vector Machines), "glmnet" (Elastic-net Generalized Linear Model), and "pam" (Prediction Analysis of Microarrays)

k.folds	How many and what fractions of dataset held-out for prediction (i.e. 3 = 1/3, 10 = 1/10, etc.)
metric	Performance metric to assess. Available options are "Accuracy", "Kappa", and "ROC.AUC".
nperm	Number of permutations, default nperm = 10
allowParallel	Logical argument dictating if parallel processing is allowed via foreach package. Default allowParallel = FALSE
create.plot	Logical argument whether to create a distribution plot of permuation results.
verbose	Logical argument whether output printed automatically in 'pretty' format. Default create.plot = FALSE
...	Extra arguments that the user would like to apply to the models

Value

p.value	Resulting p-value of permuation test

Author(s)

Charles Determan Jr.

References

Guo Y., et. al. (2010) *Sample size and statistical power considerations in high-dimensionality data settings: a comparative study of classification algorithms*. BMC Bioinformatics 11:447.

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')

perm.class(fits, vars, groups, "rf", k.folds=5,
           metric="Accuracy", nperm=10)
```

perm.features *Feature Selection via Monte Carlo Permutation*

Description

Applies Monte Carlo permutations to user specified models. The user can either use the results from `fs.stability` or provide specified model parameters.

Usage

```
perm.features(fs.model = NULL, X, Y, method, sig.level = 0.05, nperm = 10,
  allowParallel = FALSE, verbose = TRUE, ...)
```

Arguments

`fs.model`	Object containing results from `fs.stability`
`X`	A scaled matrix or dataframe containing numeric values of each feature
`Y`	A factor vector containing group membership of samples
`method`	A vector listing models to be fit. Available options are `"plsda"` (Partial Least Squares Discriminant Analysis), `"rf"` (Random Forest), `"gbm"` (Gradient Boosting Machine), `"svm"` (Support Vector Machines), `"glmnet"` (Elastic-net Generalized Linear Model), and `"pam"` (Prediction Analysis of Microarrays)
`sig.level`	Desired significance level for features, default `sig.level = .05`
`nperm`	Number of permutations, default `nperm = 10`
`allowParallel`	Logical argument dictating if parallel processing is allowed via foreach package. Default `allowParallel = FALSE`
`verbose`	Logical argument whether output printed automatically in 'pretty' format.
`...`	Extra arguments that the user would like to apply to the models

Value

`sig.level`	User-specified significance level
`num.sig.features`	Number of significant features
`sig.features`	Dataframe of significant features

Author(s)

Charles Determan Jr.

References

Wongravee K., et. al. (2009) *Monte-Carlo methods for determining optimal number of significant variables. Application to mouse urinary profiles*. Metabolomics 5:387-406.

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')

# permute variables/features
perm.features(fits, vars, groups, "rf",
              sig.level = .05, nperm = 10)
```

---

pof *Percentage of Overlapping Features*

Description

Calculates percent of overlapping features between two vectors of features. In brief, the closer to 1 the more similar the vectors. The two vectors may have an arbitrary cardinality (i.e. don't need same length).

Usage

```
pof(x, y)
```

Arguments

x	Character vector of feature names
y	Character vector of feature names

Value

Returns the percent of overlapping features for the two vectors. It takes values in [0,1], with 0 meaning no overlap between two sets and 1 meaning two sets are identical.

Author(s)

Charles E. Determan Jr.

References

Shi L., et al. (2005) *Cross-platform comparability of microarray technology: intra-platform consistency and appropriate data analysis procedures are essential*. BMC Bioinformatics. 6 (Suppl. 2) S12. He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

kuncheva, sorensen, ochiai, pof, pairwise.stability, pairwise.model.stability

Examples

```
# Percent-Overlapping Features demo
v1 <- paste("Metabolite", seq(10), sep="_")
v2 <- sample(v1, 10)
pof(v1, v2)
```

---

predicting *Model Group Prediction*

Description

This function evaluates a single fitted model and returns the predicted group memberships.

Usage

```
predicting(method, modelFit, orig.data, indicies, newdata, param = NULL)
```

Arguments

method	String of the model to be evaluated
modelFit	The fitted model being evaluated
orig.data	The orginal data before subsetting training sets. Required to have the 'observed' group membership
indicies	The indicies for the training subsets
newdata	The testing data to predict group membership
param	The parameters being fit to the model (Determined by model optimization).

Value

Returns a list of predicted group membership

prediction.metrics *Prediction Metric Calculations*

Description

Performance evaluation of all fitted models. This function concisely provides model performance metrics, including confusion matrix and ROC.

Usage

```
prediction.metrics(finalModel, method, raw.data, inTrain, outTrain, features,
  bestTune, grp.levs, stability.metric)
```

Arguments

finalModel	List of fitted models
method	Vector of strings dictating the models that were fit
raw.data	Original dataset prior to any training subset
inTrain	List of training indicies for each feature selection run
outTrain	List of testing data indicies for each feature selection run
features	List of selected features for each model
bestTune	List of parameters that have been optimized for the each respective model
grp.levs	Vector of group levels
stability.metric	A character object specifying the stability metric

Value

Returns a dataframe consisting of each feature selection runs evaluated Accuracy, Kappa, ROC.AUC, Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value.

See Also

performance.stats, perf.calc caret function confusionMatrix

predictNewClasses *Class Prediction*

Description

This function evaluates a single fitted model and returns the predicted group memberships of new data.

Usage

```
predictNewClasses(modelFit, method, orig.data, newdata, param = NULL)
```

Arguments

modelFit	The fitted model being evaluated
method	String of the model to be evaluated
orig.data	The orginal data before subsetting training sets. Required to have the 'observed' group membership
newdata	The testing data to predict group membership
param	Optional alternate parameters being fit to the model

Value

Returns a list of predicted group membership

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

fits <- fs.stability(vars,
                     groups,
                     method = c("plsda", "rf"),
                     f = 10,
                     k = 3,
                     k.folds = 10,
                     verbose = 'none')

newdata <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)$discr.mat

orig.df <- data.frame(vars, groups)

# see what the PLSDA predicts for the new data
# NOTE, newdata does not require a .classes column
predictNewClasses(fits, "plsda", orig.df, newdata)
```

RPT *Robustness-Performance Trade-Off*

Description

A variation on the F-measure (precision and recall) to assess robustness versus classification performance.

Usage

```
RPT(stability, performance, beta = 1)
```

Arguments

stability	Stability metric i.e. result from jaccard, sorensen, etc.
performance	Model performance e.g. accuracy
beta	Relative of importance of stability versus performance. Default beta = 1 treats stability and performance equally.

Value

Harmonic mean of robustness and classification performance

References

Saeys Y., Abeel T., et. al. (2008) *Machine Learning and Knowledge Discovery in Databases*. 313-325. http://link.springer.com/chapter/10.1007/978-3-540-87481-2_21

Examples

```
# RPT demo
RPT(stability=0.85, performance=0.90, beta=1)
```

sequester *Sequester Additional Parameters*

Description

When the user provides additional arguments to either fs.stability or fs.ensembl.stability this function will extract the parameters to be fit if optimization is not used i.e. optimize = FALSE.

Usage

```
sequester(theDots, method)
```

Arguments

theDots	List of additional arguments
method	Vector of strings listing models to be fit

Value

Returns a list of the following elements

parameters	The parameters that will be fit to models
pnames	The names of the specific parameters

---

sorensen *Dice-Sorensen's Index*

Description

Calculates Dice-Sorensen's index between two vectors of features. In brief, the closer to 1 the more similar the vectors. The two vectors may have an arbitrary cardinality (i.e. don't need same length). Very similar to the Jaccard Index jaccard but Dice-Sorensen is the harmonic mean of the ratio.

Usage

```
sorensen(x, y)
```

Arguments

x	vector of feature names
y	vector of feature names

Value

Returns the Dice-Sorensen's Index for the two vectors. It takes values in [0,1], with 0 meaning no overlap between two sets and 1 meaning two sets are identical.

Author(s)

Charles E. Determan Jr.

References

Sorensen T. (1948) *A method of establishing roups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons*. Kongelige Danske Videnskabernes Selskab. 5(4): 1-34.

Dice, Lee R. (1945) *Measures of the Amount of Ecologic Association Between Species*. Ecology 26 (3): 297-302. doi:10.2307/1932409

He. Z. & Weichuan Y. (2010) *Stable feature selection for biomarker discovery*. Computational Biology and Chemistry 34 215-225.

See Also

kuncheva, sorensen, ochiai, pof, pairwise.stability, pairwise.model.stability

Examples

```
# Dice-Sorensen demo
v1 <- paste("Metabolite", seq(10), sep="_")
v2 <- sample(v1, 10)
sorensen(v1, v2)
```

spearman *Spearman Rank Correlation Coefficient*

Description

Calculates spearman rank correlation between two vectors

Usage

```
spearman(x, y)
```

Arguments

x	numeric vector of ranks
y	numeric vector of ranks with compatible length to x

Value

Returns the spearman rank coefficient for the two vectors

Examples

```
# Spearman demo
v1 <- seq(10)
v2 <- sample(v1, 10)
spearman(v1, v2)
```

svm.weights *SVM Multiclass Weights Ranking*

Description

This calculates feature weights for multiclass Support Vector Machine (SVM) problems

Usage

```
## S3 method for class 'weights'
svm(model)
```

Arguments

model	A fitted SVM model of multiclass

Value

Vector of feature weights

References

Guyon I. et. al. (2010) *Gene Selection for Cancer Classification using Support Vector Machines*. Machine Learning 46 389-422.

`svmrfeFeatureRanking` *SVM Recursive Feature Extraction (Binary)*

Description

This conducts feature selection for Support Vector Machines models via recursive feature extraction. This returns a vector of the features in x ordered by relevance. The first item of the vector has the index of the feature which is more relevant to perform the classification and the last item of the vector has the feature which is less relevant. This function is specific to Binary classification problems,

Usage

```
svmrfeFeatureRanking(x, y, c, perc.rem = 10)
```

Arguments

`x`	A matrix where each column represents a feature and each row represents a sample
`y`	A vector of labels corresponding to each sample's group membership
`c`	A numeric value corresponding to the 'cost' applied during the svm model fitting. This can be selected by the user if using this function directly or is done internally.
`perc.rem`	A numeric value indicating the percent of features removed during each iteration. Default `perc.rem = 10`.

Value

Vector of features ranked from most important to least important.

References

Guyon I. et. al. (2010) *Gene Selection for Cancer Classification using Support Vector Machines*. Machine Learning 46 389-422.

See Also

`svmrfeFeatureRankingForMulticlass`

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes
```

```
# binary class feature ranking
svmrfeFeatureRanking(x = vars,
                     y = groups,
                     c = 0.1,
                     perc.rem = 10)
```

`svmrfeFeatureRankingForMulticlass` *SVM Recursive Feature Extraction (Multiclass)*

Description

This conducts feature selection for Support Vector Machines models via recursive feature extraction. This returns a vector of the features in x ordered by relevance. The first item of the vector has the index of the feature which is more relevant to perform the classification and the last item of the vector has the feature which is less relevant. This function is specific to Binary classification problems.

Usage

```
svmrfeFeatureRankingForMulticlass(x, y, c, perc.rem = 10)
```

Arguments

`x`	A matrix where each column represents a feature and each row represents a sample
`y`	A vector of labels corresponding to each sample's group membership
`c`	A numeric value corresponding to the 'cost' applied during the svm model fitting. This can be selected by the user if using this function directly or is done internally.
`perc.rem`	A numeric value indicating the percent of features removed during each iteration. Default `perc.rem = 10`.

Value

Vector of features ranked from most important to least important.

References

Guyon I. et. al. (2010) *Gene Selection for Cancer Classification using Support Vector Machines*. Machine Learning 46 389-422.

See Also

`svmrfeFeatureRanking`

Examples

```
dat.discr <- create.discr.matrix(
    create.corr.matrix(
        create.random.matrix(nvar = 50,
                             nsamp = 100,
                             st.dev = 1,
                             perturb = 0.2)),
    D = 10,
    num.groups=4
)

vars <- dat.discr$discr.mat
groups <- dat.discr$classes

# multiclass
svmrfeFeatureRankingForMulticlass(x = vars,
                                  y = groups,
                                  c = 0.1,
                                  perc.rem = 10)
```

`training` *Model Training*

Description

This fits each model with the defined parameters

Usage

```
training(data, method, tuneValue, obsLevels, theDots = NULL)
```

Arguments

`data`	Dataframe consisting of both numeric feature values and a single column named '.classes' to denoted group membership.
`method`	String dictating which model to fit
`tuneValue`	List of parameters to be applied to the specific model
`obsLevels`	Observed group levels
`theDots`	List of additional parameters to be applied to the specific model

Value

`fit` Fitted model with list with the following elements:

- xNames: Names of the features
- tuneValue: Parameters applied to the fitted model
- obsLevels: Observed levels of the groups

Author(s)

Charles Determan Jr

`tune.instructions` *Model Optimization Instructions*

Description

Provides directions for which parameters to loop over during tuning. This becomes important when certain models can access 'lower' parameters without running them independently.

Usage

```
tune.instructions(method, grid)
```

Arguments

`method`	Vector of strings indicating which models will be fit
`grid`	A list of parameters grids to be applied to the models

Value

`modelInfo` List of the following components

.

- scheme: String dictating which looping scheme to apply
- loop: Dataframe of parameters to loop through for each model
- model: Information regarding parameters of specific model
- constant: Names of the 'loop' dataframe components
- vary: Indication of parameters that vary and can access recursively

Author(s)

Charles E. Determan Jr.

Index