Package 'eclust'

July 22, 2025

Type Package

Title Environment Based Clustering for Interpretable Predictive Models in High Dimensional Data

Version 0.1.0

Description Companion package to the paper: An analytic approach for interpretable predictive models in high dimensional data, in the presence of interactions with exposures. Bhatnagar, Yang, Khundrakpam, Evans, Blanchette, Bouchard, Greenwood (2017) <DOI:10.1101/102475>. This package includes an algorithm for clustering high dimensional data that can be affected by an environmental factor.

Depends R (>= 3.3.1)

License MIT + file LICENSE

URL https://github.com/sahirbhatnagar/eclust/, http://sahirbhatnagar.com/eclust/

BugReports https://github.com/sahirbhatnagar/eclust/issues

Encoding UTF-8

LazyData true

Suggests cluster, earth, ncvreg, knitr, rmarkdown, protoclust, factoextra, ComplexHeatmap, circlize, pheatmap, viridis, pROC, glmnet

VignetteBuilder knitr

Imports caret, data.table, dynamicTreeCut, magrittr, pacman, WGCNA, stringr, pander, stats

RoxygenNote 5.0.1

NeedsCompilation no

Author Sahir Rai Bhatnagar [aut, cre] (http://sahirbhatnagar.com/)

Maintainer Sahir Rai Bhatnagar <sahir.bhatnagar@gmail.com>

Repository CRAN

Date/Publication 2017-01-26 12:08:12

Contents

`plot.eclust` *Plot Heatmap of Cluster Summaries by Exposure Status*

Description

Plots cluster summaries such as the 1st principal component or average by exposure status. This is a plot method for object of class eclust returned by the `r_cluster_data` function. Two heatmaps, side-by-side are returned, where the first heatmap corresponds to the unexposed subjects and the second heatmap corresponds to the exposed subjects.

Usage

```
## S3 method for class 'eclust'
plot(x, type = c("ECLUST", "CLUST"), summary = c("pc",
  "avg"), sample = c("training", "test"), unexposed_title = "E=0",
  exposed_title = "E=1", ...)
```

Arguments

`x`	object of class `eclust`, which is returned by the `r_cluster_data` function
`type`	show results from the "ECLUST" (which considers the environment) or "CLUST" (which ignores the environment) methods. Default is "ECLUST". See `r_cluster_data` for details. This function uses the `clustersAddon` object for "ECLUST" and the `clustersAll` for "CLUST"

summary	show the 1st principal component or the average of each cluster. Default is "pc".
sample	which sample to show, the "training" or the "test" set. Default is "training". This is determined by the train_index and test_index arguments in the r_cluster_data function. If you want to show all subjects, then provide the numeric vector 1:n to either argument, where n is the entire sample size.
unexposed_title	The title for the unexposed subjects heatmap. Default is "E=0".
exposed_title	The title for the exposed subjects heatmap. Default is "E=1".
...	other arguments passed to the Heatmap function

Details

Rows are the cluster summaries and columns are the subjects. This function determines the minimum and maximum value for the whole dataset and then creates a color scale using those values with the colorRamp2. This is so that both heatmaps are on the same color scale, i.e., each color represents the same value in both heatmaps. This is done for being able to visually compare the results.

Value

a plot of two Heatmaps, side-by-side, of the cluster summaries by exposure status

Examples

```
## Not run:
data("tcgaov")
tcgaov[1:5,1:6, with = FALSE]
Y <- log(tcgaov[["OS"]])
E <- tcgaov[["E"]]
genes <- as.matrix(tcgaov[,-c("OS","rn","subtype","E","status"),with = FALSE])
trainIndex <- drop(caret::createDataPartition(Y, p = 1, list = FALSE, times = 1))
testIndex <- setdiff(seq_len(length(Y)),trainIndex)

cluster_res <- r_cluster_data(data = genes,
                              response = Y,
                              exposure = E,
                              train_index = trainIndex,
                              test_index = testIndex,
                              cluster_distance = "tom",
                              eclust_distance = "difftom",
                              measure_distance = "euclidean",
                              clustMethod = "hclust",
                              cutMethod = "dynamic",
                              method = "average",
                              nPC = 1,
                              minimum_cluster_size = 60)

class(cluster_res)

plot(cluster_res, show_column_names = FALSE)
```

```
## End(Not run)
```

`plot.similarity` *Function to generate heatmap*

Description

Plots a heatmap of a similarity matrix such as a correlation matrix or a TOM matrix. This function is a plotting method for an object of class similarity. These objects are returned by the `s_generate_data` and `s_generate_data_mars` functions

Usage

```
## S3 method for class 'similarity'
plot(x, color = viridis::viridis(100), truemodule,
  active, ...)
```

Arguments

`x`	an object of class similarity. This is a p x p symmetric matix such as a correlation matrix or a TOM matrix, where p is the number of genes
`color`	colors for the heatmap. By default it uses the `viridis` color scheme. The `viridis` package needs to be installed.
`truemodule`	a numeric vector of length p where p is the number of genes, giving the module membership. By default, 0 = Grey, 1 = Turquoise, 2 = Blue, 3 = Red, 4 = Green, and 5 = Yellow. This information is used for annotating the heatmap
`active`	a binary vector of length p (where p is the number of genes) where 0 means that gene is not related to the response, and 1 means that the gene is associated to the response.
`...`	other arguments passed to the pheatmap function

Value

a heatmap of a similarity matrix

Note

this function is only meant to be used with output from the `s_generate_data` and `s_generate_data_mars` functions, since it assumes a fixed number of modules.

Examples

```
## Not run:
corrX <- cor(simdata[,c(-1,-2)])
class(corrX) <- append(class(corrX), "similarity")
plot(corrX, truemodule = c(rep(1:5, each=150), rep(0, 250)))

## End(Not run)
```

`r_cluster_data` *Cluster data using environmental exposure*

Description

This is one of the functions for real data analysis, which will cluster the data based on the environment, as well as ignoring the environment

Usage

```
r_cluster_data(data, response, exposure, train_index, test_index,
  cluster_distance = c("corr", "corr0", "corr1", "tom", "tom0", "tom1",
  "diffcorr", "difftom", "fisherScore"), eclust_distance = c("fisherScore",
  "corScor", "diffcorr", "difftom"), measure_distance = c("euclidean",
  "maximum", "manhattan", "canberra", "binary", "minkowski"),
  minimum_cluster_size = 50, ...)
```

Arguments

`data`	n x p matrix of data. rows are samples, columns are genes or cpg sites. Should not contain the environment variable
`response`	numeric vector of length n
`exposure`	binary (0,1) numeric vector of length n for the exposure status of the n samples
`train_index`	numeric vector indcating the indices of `response` and the rows of `data` that are in the training set
`test_index`	numeric vector indcating the indices of `response` and the rows of `data` that are in the test set
`cluster_distance`	character representing which matrix from the training set that you want to use to cluster the genes. Must be one of the following • corr, corr0, corr1, tom, tom0, tom1, diffcorr, difftom, corScor, tomScor, fisherScore
`eclust_distance`	character representing which matrix from the training set that you want to use to cluster the genes based on the environment. See `cluster_distance` for avaialble options. Should be different from `cluster_distance`. For example, if `cluster_distance=corr` and `EclustDistance=fisherScore`. That is, one should be based on correlations ignoring the environment, and the other should be based on correlations accounting for the environment. This function will always return this add on
`measure_distance`	one of "euclidean","maximum","manhattan", "canberra", "binary","minkowski" to be passed to `dist` function for calculating the distance for the clusters based on the corr,corr1,corr0, tom, tom0, tom1 matrices

`minimum_cluster_size`
: The minimum cluster size. Only applicable if `cutMethod='dynamic'`. This argument is passed to the `cutreeDynamic` function through the `u_cluster_similarity` function. Default is 50.

`...`
: arguments passed to the `u_cluster_similarity` function

Details

This function clusters the data. The results of this function should then be passed to the `r_prepare_data` function which output the appropriate X and Y matrices in the right format for regression packages such as `mgcv`, `caret` and `glmnet`

Value

a list of length 8:

clustersAddon clustering results based on the environment and not the environment. see `u_cluster_similarity` for details

clustersAll clustering results ignoring the environment. See `u_cluster_similarity` for details

etrain vector of the exposure variable for the training set

cluster_distance_similarity the similarity matrix based on the argument specified in `cluster_distance`

eclust_distance_similarity the similarity matrix based on the argument specified in `eclust_distance`

clustersAddonMembership a data.frame and data.table of the clustering membership for clustering results based on the environment and not the environment. As a result, each gene will show up twice in this table

clustersAllMembership a data.frame and data.table of the clustering membership for clustering results based on all subjects i.e. ignoring the environment. Each gene will only show up once in this table

clustersEclustMembership a data.frame and data.table of the clustering membership for clustering results accounting for the environment. Each gene will only show up once in this table

See Also

`u_cluster_similarity`

Examples

```
data("tcgaov")
tcgaov[1:5,1:6, with = FALSE]
Y <- log(tcgaov[["OS"]])
E <- tcgaov[["E"]]
genes <- as.matrix(tcgaov[,-c("OS","rn","subtype","E","status"),with = FALSE])
trainIndex <- drop(caret::createDataPartition(Y, p = 0.5, list = FALSE, times = 1))
testIndex <- setdiff(seq_len(length(Y)),trainIndex)

## Not run:
cluster_res <- r_cluster_data(data = genes,
                              response = Y,
```

```
                                exposure = E,
                                train_index = trainIndex,
                                test_index = testIndex,
                                cluster_distance = "tom",
                                eclust_distance = "difftom",
                                measure_distance = "euclidean",
                                clustMethod = "hclust",
                                cutMethod = "dynamic",
                                method = "average",
                                nPC = 1,
                                minimum_cluster_size = 60)

# the number of clusters determined by the similarity matrices specified
# in the cluster_distance and eclust_distance arguments. This will always be larger
# than cluster_res$clustersAll$nclusters which is based on the similarity matrix
# specified in the cluster_distance argument
cluster_res$clustersAddon$nclusters

# the number of clusters determined by the similarity matrices specified
# in the cluster_distance argument only
cluster_res$clustersAll$nclusters

## End(Not run)
```

---

r_prepare_data *Prepare data for regression routines*

Description

This function will output the appropriate X and Y matrices in the right format for regression packages such as `mgcv`, `caret` and `glmnet`

Usage

```
r_prepare_data(data, response = "Y", exposure = "E", probe_names)
```

Arguments

`data`	the data frame which contains the response, exposure, and genes or cpgs or covariates. the columns should be labelled.
`response`	the column name of the response in the `data` argument
`exposure`	the column name of the exposure in the `data` argument
`probe_names`	the column names of the genes, or cpg sites or covariates

Value

a list of length 5:

X the X matrix

Y the response vector

E the exposure vector

main_effect_names the names of the main effects including the exposure

interaction_names the names of the interaction effects

Examples

```
data("tcgaov")
tcgaov[1:5,1:6, with = FALSE]
Y <- log(tcgaov[["OS"]])
E <- tcgaov[["E"]]
genes <- as.matrix(tcgaov[,-c("OS","rn","subtype","E","status"),with = FALSE])
trainIndex <- drop(caret::createDataPartition(Y, p = 0.5, list = FALSE, times = 1))
testIndex <- setdiff(seq_len(length(Y)),trainIndex)

## Not run:
cluster_res <- r_cluster_data(data = genes,
                              response = Y,
                              exposure = E,
                              train_index = trainIndex,
                              test_index = testIndex,
                              cluster_distance = "tom",
                              eclust_distance = "difftom",
                              measure_distance = "euclidean",
                              clustMethod = "hclust",
                              cutMethod = "dynamic",
                              method = "average",
                              nPC = 1,
                              minimum_cluster_size = 50)

pc_eclust_interaction <- r_prepare_data(data = cbind(cluster_res$clustersAddon$PC,
                                                     survival = Y[trainIndex],
                                                     subtype = E[trainIndex]),
                                        response = "survival", exposure = "subtype")
names(pc_eclust_interaction)
dim(pc_eclust_interaction$X)
pc_eclust_interaction$main_effect_names
pc_eclust_interaction$interaction_names

## End(Not run)
```

simdata *Simulated Data with Environment Dependent Correlations*

Description

A dataset containing simulated data for example use of the `eclust` package functions. This data was generated using the `s_modules` and `s_generate_data`

Usage

```
simdata
```

Format

A matrix with 100 rows and 502 variables:

Y continuous response vector

E binary environment variable for ECLUST method. E = 0 for unexposed (n=50) and E = 1 for exposed (n=50)

columns 3:502 gene expression data for 1000 genes. column names are the gene names

Note

Code used to generate this data can be found on the GitHub page for this package. See URL below.

Source

https://raw.githubusercontent.com/sahirbhatnagar/eclust/master/data-raw/simulated-data-processing.R

References

Bhatnagar, SR., Yang, Y., Blanchette, M., Bouchard, L., Khundrakpam, B., Evans, A., Greenwood, CMT. (2016+). *An analytic approach for interpretable predictive models in high dimensional data, in the presence of interactions with exposures* Preprint

Examples

```
simdata[1:5, 1:10]
table(simdata[,"E"])
```

s_generate_data *Generate linear response data and test and training sets for simulation study*

Description

create a function that takes as input, the number of genes, the true beta vector, the gene expression matrix created from the generate_blocks function and returns a list of data matrix, as well as correlation matrices, TOM matrices, cluster information, training and test data

Usage

```
s_generate_data(p, X, beta, binary_outcome = FALSE,
  cluster_distance = c("corr", "corr0", "corr1", "tom", "tom0", "tom1",
  "diffcorr", "difftom", "corScor", "tomScor", "fisherScore"), n, n0,
  include_interaction = F, signal_to_noise_ratio = 1,
  eclust_distance = c("fisherScore", "corScor", "diffcorr", "difftom"),
  cluster_method = c("hclust", "protoclust"), cut_method = c("dynamic",
  "gap", "fixed"), distance_method = c("euclidean", "maximum", "manhattan",
  "canberra", "binary", "minkowski"), n_clusters,
  agglomeration_method = c("complete", "average", "ward.D2", "single",
  "ward.D", "mcquitty", "median", "centroid"), nPC = 1, K.max = 10,
  B = 10)
```

Arguments

p	number of genes in design matrix
X	gene expression matrix of size n x p using the generate_blocks function
beta	true beta coefficient vector
binary_outcome	Logical. Should a binary outcome be generated. Default is FALSE. See details on how a binary outcome is generated
cluster_distance	character representing which matrix from the training set that you want to use to cluster the genes. Must be one of the following • corr, corr0, corr1, tom, tom0, tom1, diffcorr, difftom, corScor, tomScor, fisherScore
n	total number of subjects
n0	total number of subjects with E=0
include_interaction	Should an interaction with the environment be generated as part of the response. Default is FALSE.
signal_to_noise_ratio	signal to noise ratio, default is 1

`eclust_distance`
: character representing which matrix from the training set that you want to use to cluster the genes based on the environment. See `cluster_distance` for avaialble options. Should be different from `cluster_distance`. For example, if `cluster_distance=corr` and `EclustDistance=fisherScore`. That is, one should be based on correlations ignoring the environment, and the other should be based on correlations accounting for the environment. This function will always return this add on

`cluster_method`
: Cluster the data using hierarchical clustering or prototype clustering. Defaults `cluster_method="hclust"`. Other option is `protoclust`, however this package must be installed before proceeding with this option

`cut_method`
: what method to use to cut the dendrogram. `'dynamic'` refers to `dynamicTreeCut` library. `'gap'` is Tibshirani's gap statistic `clusGap` using the `'Tibs2001SEmax'` rule. `'fixed'` is a fixed number specified by the `n_clusters` argument

`distance_method`
: one of "euclidean","maximum","manhattan", "canberra", "binary","minkowski" to be passed to `dist` function.

`n_clusters`
: Number of clusters specified by the user. Only applicable when `cut_method="fixed"`

`agglomeration_method`
: the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

`nPC`
: number of principal components to extract from each cluster. Default is 1. Only 1 or 2 is allowed.

`K.max`
: the maximum number of clusters to consider, must be at least two. Only used if `cutMethod='gap'`

`B`
: integer, number of Monte Carlo ("bootstrap") samples. Only used if `cutMethod='gap'`

Details

To generate a binary outcome we first generate a continuous outcome Y which is $X^T\beta$, defined $p = 1/(1 + exp(-Y))$ and used this to generate a two-class outcome z with $Pr(z = 1) = p$ and $Pr(z = 0) = 1 - p$.

Value

list of (in the following order)

beta_truth a 1 column matrix containing the true beta coefficient vector

similarity an object of class similarity which is the similarity matrix specified by the `cluster_distance` argument

similarityEclust an object of class similarity which is the similarity matrix specified by the `eclust_distance` argument

DT data.table of simulated data from the `s_response` function

Y The simulated response

X0 the n0 x p design matrix for the unexposed subjects

X1 the n1 x p design matrix for the exposed subjects

X_train the training design matrix for all subjects

X_test the test set design matrix for all subjects

Y_train the training set response

Y_test the test set response

DT_train the training response and training design matrix in a single data.frame object

DT_test the test response and training design matrix in a single data.frame object

S0 a character vector of the active genes i.e. the ones that are associated with the response

n_clusters_All the number of clusters identified by using the similarity matrix specified by the `cluster_distance` argument

n_clusters_Eclust the number of clusters identified by using the similarity matrix specified by the `eclust_distance` argument

n_clusters_Addon the sum of `n_clusters_All` and `n_clusters_Eclust`

clustersAll the cluster membership of each gene based on the `cluster_distance` matrix

clustersAddon the cluster membership of each gene based on both the `cluster_distance` matrix and the `eclust_distance` matrix. Note that each gene will appear twice here

clustersEclust the cluster membership of each gene based on the `eclust_distance` matrix

gene_groups_inter cluster membership of each gene with a penalty factor used for the group lasso

gene_groups_inter_Addon cluster membership of each gene with a penalty factor used for the group lasso

tom_train_all the TOM matrix based on all training subjects

tom_train_diff the absolute difference of the exposed and unexposed TOM matrices: $|TOM_{E=1} - TOM_{E=0}|$

tom_train_e1 the TOM matrix based on training exposed subjects only

tom_train_e0 the TOM matrix based on training unexposed subjects only

corr_train_all the Pearson correlation matrix based on all training subjects

corr_train_diff the absolute difference of the exposed and unexposed Pearson correlation matrices: $|Cor_{E=1} - Cor_{E=0}|$

corr_train_e1 the Pearson correlation matrix based on training exposed subjects only

corr_train_e0 the Pearson correlation matrix based on training unexposed subjects only

fisherScore The fisher scoring matrix. see `u_fisherZ` for details

corScor The correlation scoring matrix: $|Cor_{E=1} + Cor_{E=0} - 2|$

mse_null The MSE for the null model

DT_train_folds The 10 training folds used for the stability measures

X_train_folds The 10 X training folds (the same as in DT_train_folds)

Y_train_folds The 10 Y training folds (the same as in DT_train_folds)

Note

this function calls the s_response to generate phenotype as a function of the gene expression data. This function also returns other information derived from the simulated data including the test and training sets, the correlation and TOM matrices and the clusters.

the PCs and averages need to be calculated in the fitting functions, because these will change based on the CV fold

Examples

```
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)
betaMainInteractions <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
```

```
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
betaMainInteractions[which(betaMainEffect!=0)] <- runif(nActive, alphaMean - 0.1, alphaMean + 0.1)
beta <- c(betaMainEffect, betaE, betaMainInteractions)
## Not run:
result <- s_generate_data(p = p, X = X,
                          beta = beta,
                          include_interaction = TRUE,
                          cluster_distance = cluster_distance,
                          n = n, n0 = n0,
                          eclust_distance = Ecluster_distance,
                          signal_to_noise_ratio = SNR,
                          distance_method = distanceMethod,
                          cluster_method = clustMethod,
                          cut_method = cutMethod,
                          agglomeration_method = agglomerationMethod,
                          nPC = 1)
names(result)

## End(Not run)
```

`s_generate_data_mars` *Generate non linear response and test and training sets for non-linear simulation study*

Description

create a function that takes as input, the number of genes, the true beta vector, the gene expression matrix created from the generate_blocks function and returns a list of data matrix, as well as correlation matrices, TOM matrices, cluster information, training and test data

Usage

```
s_generate_data_mars(p, X, beta, binary_outcome = FALSE, truemodule, nActive,
  cluster_distance = c("corr", "corr0", "corr1", "tom", "tom0", "tom1",
  "diffcorr", "difftom", "corScor", "tomScor", "fisherScore"), n, n0,
  include_interaction = F, signal_to_noise_ratio = 1,
  eclust_distance = c("fisherScore", "corScor", "diffcorr", "difftom"),
  cluster_method = c("hclust", "protoclust"), cut_method = c("dynamic",
  "gap", "fixed"), distance_method = c("euclidean", "maximum", "manhattan",
  "canberra", "binary", "minkowski"), n_clusters,
  agglomeration_method = c("complete", "average", "ward.D2", "single",
  "ward.D", "mcquitty", "median", "centroid"), nPC = 1, K.max = 10,
  B = 10)
```

Arguments

p	number of genes in design matrix
X	gene expression matrix of size n x p using the generate_blocks function
beta	true beta coefficient vector
binary_outcome	Logical. Should a binary outcome be generated. Default is FALSE. See details on how a binary outcome is generated
truemodule	numeric vector of the true module membership used in the s_response_mars function. Modules 3 and 4 are active in the response. See s_response_mars function for details.
nActive	number of active genes in the response used in the s_response_mars
cluster_distance	character representing which matrix from the training set that you want to use to cluster the genes. Must be one of the following • corr, corr0, corr1, tom, tom0, tom1, diffcorr, difftom, corScor, tomScor, fisherScore
n	total number of subjects
n0	total number of subjects with E=0
include_interaction	Should an interaction with the environment be generated as part of the response. Default is FALSE.
signal_to_noise_ratio	signal to noise ratio, default is 1
eclust_distance	character representing which matrix from the training set that you want to use to cluster the genes based on the environment. See cluster_distance for avaialble options. Should be different from cluster_distance. For example, if cluster_distance=corr and EclustDistance=fisherScore. That is, one should be based on correlations ignoring the environment, and the other should be based on correlations accounting for the environment. This function will always return this add on
cluster_method	Cluster the data using hierarchical clustering or prototype clustering. Defaults cluster_method="hclust". Other option is protoclust, however this package must be installed before proceeding with this option
cut_method	what method to use to cut the dendrogram. 'dynamic' refers to dynamicTreeCut library. 'gap' is Tibshirani's gap statistic clusGap using the 'Tibs2001SEmax' rule. 'fixed' is a fixed number specified by the n_clusters argument
distance_method	one of "euclidean","maximum","manhattan", "canberra", "binary","minkowski" to be passed to dist function.
n_clusters	Number of clusters specified by the user. Only applicable when cut_method="fixed"
agglomeration_method	the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

nPC	number of principal components. Can be 1 or 2.
K.max	the maximum number of clusters to consider, must be at least two. Only used if cutMethod='gap'
B	integer, number of Monte Carlo ("bootstrap") samples. Only used if cutMethod='gap'

Value

list of (in the following order)

beta_truth a 1 column matrix containing the true beta coefficient vector

similarity an object of class similarity which is the similarity matrix specified by the cluster_distance argument

similarityEclust an object of class similarity which is the similarity matrix specified by the eclust_distance argument

DT data.table of simulated data from the s_response function

Y The simulated response

X0 the n0 x p design matrix for the unexposed subjects

X1 the n1 x p design matrix for the exposed subjects

X_train the training design matrix for all subjects

X_test the test set design matrix for all subjects

Y_train the training set response

Y_test the test set response

DT_train the training response and training design matrix in a single data.frame object

DT_test the test response and training design matrix in a single data.frame object

S0 a character vector of the active genes i.e. the ones that are associated with the response

n_clusters_All the number of clusters identified by using the similarity matrix specified by the cluster_distance argument

n_clusters_Eclust the number of clusters identified by using the similarity matrix specified by the eclust_distance argument

n_clusters_Addon the sum of n_clusters_All and n_clusters_Eclust

clustersAll the cluster membership of each gene based on the cluster_distance matrix

clustersAddon the cluster membership of each gene based on both the cluster_distance matrix and the eclust_distance matrix. Note that each gene will appear twice here

clustersEclust the cluster membership of each gene based on the eclust_distance matrix

gene_groups_inter cluster membership of each gene with a penalty factor used for the group lasso

gene_groups_inter_Addon cluster membership of each gene with a penalty factor used for the group lasso

tom_train_all the TOM matrix based on all training subjects

tom_train_diff the absolute difference of the exposed and unexposed TOM matrices: $|TOM_{E=1} - TOM_{E=0}|$

tom_train_e1 the TOM matrix based on training exposed subjects only

tom_train_e0 the TOM matrix based on training unexposed subjects only

corr_train_all the Pearson correlation matrix based on all training subjects

corr_train_diff the absolute difference of the exposed and unexposed Pearson correlation matrices: $|Cor_{E=1} - Cor_{E=0}|$

corr_train_e1 the Pearson correlation matrix based on training exposed subjects only

corr_train_e0 the Pearson correlation matrix based on training unexposed subjects only

fisherScore The fisher scoring matrix. see u_fisherZ for details

corScor The correlation scoring matrix: $|Cor_{E=1} + Cor_{E=0} - 2|$

mse_null The MSE for the null model

DT_train_folds The 10 training folds used for the stability measures

X_train_folds The 10 X training folds (the same as in DT_train_folds)

Y_train_folds The 10 Y training folds (the same as in DT_train_folds)

Examples

```
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels
```

```
X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
beta <- c(betaMainEffect, betaE)

result <- s_generate_data_mars(p = p, X = X,
                               beta = beta,
                               binary_outcome = FALSE,
                               truemodule = truemodule1,
                               nActive = nActive,
                               include_interaction = FALSE,
                               cluster_distance = cluster_distance,
                               n = n, n0 = n0,
                               eclust_distance = Ecluster_distance,
                               signal_to_noise_ratio = SNR,
                               distance_method = distanceMethod,
                               cluster_method = clustMethod,
                               cut_method = cutMethod,
                               agglomeration_method = agglomerationMethod,
                               nPC = 1)
names(result)
```

`s_mars_clust` *Fit MARS Models on Simulated Cluster Summaries*

Description

This function creates summaries of the given clusters (e.g. 1st PC or average), and then runs Friedman's MARS on those summaries. To be used with simulated data where the 'truth' is known i.e., you know which features are associated with the response. This function was used to produce the simulation results in Bhatnagar et al. 2016.

Usage

```
s_mars_clust(x_train, x_test, y_train, y_test, s0, summary = c("pc", "avg"),
  model = c("MARS"), exp_family = c("gaussian", "binomial"), gene_groups,
  true_beta = NULL, topgenes = NULL, stability = F, filter = F,
  include_E = T, include_interaction = F, clust_type = c("CLUST",
  "ECLUST"), nPC = 1)
```

Arguments

x_train	ntrain x p matrix of simulated training set where ntrain is the number of training observations and p is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
x_test	ntest x p matrix of simulated training set where ntest is the number of training observations and p is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
y_train	numeric vector of length ntrain representing the responses for the training subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class)
y_test	numeric vector of length ntest representing the responses for the test subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class).
s0	chracter vector of the active feature names, i.e., the features in x_train that are truly associated with the response.
summary	the summary of each cluster. Can be the principal component or average. Default is summary = "pc" which takes the first number_pc principal components. Currently a maximum of 2 principal components can be chosen.
model	Type of non-linear model to be fit. Currently only Friedman's MARS is supported.
exp_family	Response type. See details for y_train argument above.
gene_groups	data.frame that contains the group membership for each feature. The first column is called 'gene' and the second column should be called 'cluster'. The 'gene' column identifies the features and must be the same identifiers in the x_train, x_test matrices. The 'cluster' column is a numeric integer indicating the cluster group membership. A cluster group membership of 0 implies the feature did not cluster into any group.
true_beta	numeric vector of true beta coefficients
topgenes	List of features to keep if filter=TRUE. Default is topgenes = NULL which means all features are kept for the analysis
stability	Should stability measures be calculated. Default is stability=FALSE. See details
filter	Should analysis be run on a subset of features. Default is filter = FALSE
include_E	Should the environment variable be included in the regression analysis. Default is include_E = TRUE
include_interaction	Should interaction effects between the features in x_train and the environment variable be fit. Default is include_interaction=TRUE

`clust_type` Method used to cluster the features. This is used for naming the output only and has no consequence for the results. `clust_type = "CLUST"` is the default which means that the environment varible was not used in the clustering step. `clust_type = "ECLUST"` means that the environment variable was used in the clustering aspect.

`nPC` Number of principal components if `summary = "pc"`. Default is `nPC = 1`. Can be either 1 or 2.

Details

This function first does 10 fold cross-validation to tune the degree (either 1 or 2) using the `train` function with `method="earth"` and nprune is fixed at 1000. Then the `earth` function is used, with `nk = 1000` and `pmethod = "backward"` to fit the MARS model using the optimal degree from the 10-fold CV.

Value

This function has two different outputs depending on whether `stability = TRUE` or `stability = FALSE`

If `stability = TRUE` then this function returns a `p x 2` data.frame or data.table of regression coefficients without the intercept. The output of this is used for subsequent calculations of stability.

If `stability = FALSE` then returns a vector with the following elements (See Table 3: Measures of Performance in Bhatnagar et al (2016+) for definitions of each measure of performance):

`mse or AUC` Test set mean squared error if `exp_family = "gaussion"` or test set Area under the curve if `exp_family = "binomial"` calculated using the `roc` function

`RMSE` Square root of the mse. Only applicable if `exp_family = "gaussion"`

`Shat` Number of non-zero estimated regression coefficients. The non-zero estimated regression coefficients are referred to as being selected by the model

`TPR` true positive rate

`FPR` false positive rate

`Correct Sparsity` Correct true positives + correct true negative coefficients divided by the total number of features

`CorrectZeroMain` Proportion of correct true negative main effects

`CorrectZeroInter` Proportion of correct true negative interactions

`IncorrectZeroMain` Proportion of incorrect true negative main effects

`IncorrectZeroInter` Proportion of incorrect true negative interaction effects

Examples

```
## Not run:
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
beta <- c(betaMainEffect, betaE)

result <- s_generate_data_mars(p = p, X = X,
```

```
                                beta = beta,
                                binary_outcome = FALSE,
                                truemodule = truemodule1,
                                nActive = nActive,
                                include_interaction = FALSE,
                                cluster_distance = cluster_distance,
                                n = n, n0 = n0,
                                eclust_distance = Ecluster_distance,
                                signal_to_noise_ratio = SNR,
                                distance_method = distanceMethod,
                                cluster_method = clustMethod,
                                cut_method = cutMethod,
                                agglomeration_method = agglomerationMethod,
                                nPC = 1)


mars_res <- s_mars_clust(x_train = result[["X_train"]],
                         x_test = result[["X_test"]],
                         y_train = result[["Y_train"]],
                         y_test = result[["Y_test"]],
                         s0 = result[["S0"]],
                         summary = "pc",
                         exp_family = "gaussian",
                         gene_groups = result[["clustersAddon"]],
                         clust_type = "ECLUST")
unlist(mars_res)

## End(Not run)
```

`s_mars_separate` *Fit Multivariate Adaptive Regression Splines on Simulated Data*

Description

This function can run Friedman's MARS models on the untransformed design matrix. To be used with simulated data where the 'truth' is known i.e., you know which features are associated with the response. This function was used to produce the simulation results in Bhatnagar et al. 2016. Uses caret functions to tune the degree and the nprune parameters

Usage

```
s_mars_separate(x_train, x_test, y_train, y_test, s0, model = c("MARS"),
  exp_family = c("gaussian", "binomial"), topgenes = NULL, stability = F,
  filter = F, include_E = T, include_interaction = F, ...)
```

Arguments

`x_train`	`ntrain` x `p` matrix of simulated training set where `ntrain` is the number of training observations and `p` is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names

x_test	ntest x p matrix of simulated training set where ntest is the number of training observations and p is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
y_train	numeric vector of length ntrain representing the responses for the training subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class)
y_test	numeric vector of length ntest representing the responses for the test subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class).
s0	chracter vector of the active feature names, i.e., the features in x_train that are truly associated with the response.
model	Type of non-linear model to be fit. Currently only Friedman's MARS is supported.
exp_family	Response type. See details for y_train argument above.
topgenes	List of features to keep if filter=TRUE. Default is topgenes = NULL which means all features are kept for the analysis
stability	Should stability measures be calculated. Default is stability=FALSE. See details
filter	Should analysis be run on a subset of features. Default is filter = FALSE
include_E	Should the environment variable be included in the regression analysis. Default is include_E = TRUE
include_interaction	Should interaction effects between the features in x_train and the environment variable be fit. Default is include_interaction=TRUE
...	other parameters passed to trainControl function

Details

This function first does 10 fold cross-validation to tune the degree (either 1 or 2) using the train function with method="earth" and nprune is fixed at 1000. Then the earth function is used, with nk = 1000 and pmethod = "backward" to fit the MARS model using the optimal degree from the 10-fold CV.

Value

This function has two different outputs depending on whether stability = TRUE or stability = FALSE

If stability = TRUE then this function returns a p x 2 data.frame or data.table of regression coefficients without the intercept. The output of this is used for subsequent calculations of stability.

If stability = FALSE then returns a vector with the following elements (See Table 3: Measures of Performance in Bhatnagar et al (2016+) for definitions of each measure of performance):

mse or AUC	Test set mean squared error if exp_family = "gaussion" or test set Area under the curve if exp_family = "binomial" calculated using the roc function
RMSE	Square root of the mse. Only applicable if exp_family = "gaussion"
Shat	Number of non-zero estimated regression coefficients. The non-zero estimated regression coefficients are referred to as being selected by the model
TPR	true positive rate
FPR	false positive rate
Correct Sparsity	Correct true positives + correct true negative coefficients divided by the total number of features
CorrectZeroMain	Proportion of correct true negative main effects
CorrectZeroInter	Proportion of correct true negative interactions
IncorrectZeroMain	Proportion of incorrect true negative main effects
IncorrectZeroInter	Proportion of incorrect true negative interaction effects

Examples

```
## Not run:
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
```

```
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
beta <- c(betaMainEffect, betaE)

result <- s_generate_data_mars(p = p, X = X,
                               beta = beta,
                               binary_outcome = FALSE,
                               truemodule = truemodule1,
                               nActive = nActive,
                               include_interaction = FALSE,
                               cluster_distance = cluster_distance,
                               n = n, n0 = n0,
                               eclust_distance = Ecluster_distance,
                               signal_to_noise_ratio = SNR,
                               distance_method = distanceMethod,
                               cluster_method = clustMethod,
                               cut_method = cutMethod,
                               agglomeration_method = agglomerationMethod,
                               nPC = 1)


mars_res <- s_mars_separate(x_train = result[["X_train"]],
                            x_test = result[["X_test"]],
                            y_train = result[["Y_train"]],
                            y_test = result[["Y_test"]],
                            s0 = result[["S0"]],
                            exp_family = "gaussian")
unlist(mars_res)

## End(Not run)
```

---

s_modules *Simulate Covariates With Exposure Dependent Correlations*

Description

This is a wrapper of the simulateDatExpr function which simulates data in a modular structure (i.e. in blocks). This function simulates data in 5 blocks referred to as Turquoise, Blue, Red, Green and Yellow, separately for exposed (E=1) and unexposed (E=0) observations.

Usage

```
s_modules(n, p, rho, exposed, ...)
```

Arguments

n	number of observations
p	total number of predictors to simulate
rho	numeric value representing the expected correlation between green module and red module
exposed	binary numeric vector of length n with 0 for unexposed and 1 for exposed
...	arguments passed to the simulateDatExpr function

Value

n x p matrix of simulated data

Examples

```
library(magrittr)
p <- 1000
n <- 200
d0 <- s_modules(n = 100, p = 1000, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = 100, p = 1000, rho = 0.90, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

X <- rbind(d0$datExpr, d1$datExpr) %>%
 magrittr::set_colnames(paste0("Gene", 1:p)) %>%
 magrittr::set_rownames(paste0("Subject",1:n))
dim(X)
```

s_pen_clust *Fit Penalized Regression Models on Simulated Cluster Summaries*

Description

This function creates summaries of the given clusters (e.g. 1st PC or average), and then fits a penalized regression model on those summaries. To be used with simulated data where the 'truth' is known i.e., you know which features are associated with the response. This function was used to produce the simulation results in Bhatnagar et al. 2016. Can run lasso, elasticnet, SCAD or MCP models

Usage

```
s_pen_clust(x_train, x_test, y_train, y_test, s0, gene_groups,
  summary = c("pc", "avg"), model = c("lasso", "elasticnet", "scad", "mcp"),
  exp_family = c("gaussian", "binomial"), filter = F, topgenes = NULL,
  stability = F, include_E = T, include_interaction = F,
  clust_type = c("CLUST", "ECLUST"), number_pc = 1)
```

Arguments

x_train	ntrain x p matrix of simulated training set where ntrain is the number of training observations and p is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
x_test	ntest x p matrix of simulated training set where ntest is the number of training observations and p is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
y_train	numeric vector of length ntrain representing the responses for the training subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class)
y_test	numeric vector of length ntest representing the responses for the test subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class).
s0	chracter vector of the active feature names, i.e., the features in x_train that are truly associated with the response.
gene_groups	data.frame that contains the group membership for each feature. The first column is called 'gene' and the second column should be called 'cluster'. The 'gene' column identifies the features and must be the same identifiers in the x_train,x_test matrices. The 'cluster' column is a numeric integer indicating the cluster group membership. A cluster group membership of 0 implies the feature did not cluster into any group.

summary — the summary of each cluster. Can be the principal component or average. Default is summary = "pc" which takes the first number_pc principal components. Currently a maximum of 2 principal components can be chosen.

model — Regression model to be fit on cluster summaries. Default is model="lasso" which corresponds to glmnet mixing parameter alpha=1. model="elasticnet" corresponds to glmnet mixing parameter alpha=0.5, model="mcp" and model="scad" are the non-convex models from the ncvreg package

exp_family — Response type. See details for y_train argument above.

filter — Should analysis be run on a subset of features. Default is filter = FALSE

topgenes — List of features to keep if filter=TRUE. Default is topgenes = NULL which means all features are kept for the analysis

stability — Should stability measures be calculated. Default is stability=FALSE. See details

include_E — Should the environment variable be included in the regression analysis. Default is include_E = TRUE

include_interaction — Should interaction effects between the features in x_train and the environment variable be fit. Default is include_interaction=TRUE

clust_type — Method used to cluster the features. This is used for naming the output only and has no consequence for the results. clust_type = "CLUST" is the default which means that the environment varible was not used in the clustering step. clust_type = "ECLUST" means that the environment variable was used in the clustering aspect.

number_pc — Number of principal components if summary = "pc". Default is number_pc = 1. Can be either 1 or 2.

Details

The stability of feature importance is defined as the variability of feature weights under perturbations of the training set, i.e., small modifications in the training set should not lead to considerable changes in the set of important covariates (Toloşi, L., & Lengauer, T. (2011)). A feature selection algorithm produces a weight, a ranking, and a subset of features. In the CLUST and ECLUST methods, we defined a predictor to be non-zero if its corresponding cluster representative weight was non-zero. Using 10-fold cross validation (CV), we evaluated the similarity between two features and their rankings using Pearson and Spearman correlation, respectively. For each CV fold we re-ran the models and took the average Pearson/Spearman correlation of the 10 choose 2 combinations of estimated coefficients vectors. To measure the similarity between two subsets of features we took the average of the Jaccard distance in each fold. A Jaccard distance of 1 indicates perfect agreement between two sets while no agreement will result in a distance of 0.

Value

This function has two different outputs depending on whether stability = TRUE or stability = FALSE

If stability = TRUE then this function returns a p x 2 data.frame or data.table of regression coefficients without the intercept. The output of this is used for subsequent calculations of stability.

If stability = FALSE then returns a vector with the following elements (See Table 3: Measures of Performance in Bhatnagar et al (2016+) for definitions of each measure of performance):

mse or AUC	Test set mean squared error if exp_family = "gaussion" or test set Area under the curve if exp_family = "binomial" calculated using the roc function
RMSE	Square root of the mse. Only applicable if exp_family = "gaussion"
Shat	Number of non-zero estimated regression coefficients. The non-zero estimated regression coefficients are referred to as being selected by the model
TPR	true positive rate
FPR	false positive rate
Correct Sparsity	Correct true positives + correct true negative coefficients divided by the total number of features
CorrectZeroMain	Proportion of correct true negative main effects
CorrectZeroInter	Proportion of correct true negative interactions
IncorrectZeroMain	Proportion of incorrect true negative main effects
IncorrectZeroInter	Proportion of incorrect true negative interaction effects
nclusters	number of estimated clusters by the cutreeDynamic function

Note

number_pc=2 will not work if there is only one feature in an estimated cluster

References

Tološi, L., & Lengauer, T. (2011). *Classification with correlated features: unreliability of feature ranking and solutions. Bioinformatics, 27(14), 1986-1994.*

Bhatnagar, SR., Yang, Y., Blanchette, M., Bouchard, L., Khundrakpam, B., Evans, A., Greenwood, CMT. (2016+). *An analytic approach for interpretable predictive models in high dimensional data, in the presence of interactions with exposures Preprint*

Langfelder, P., Zhang, B., & Horvath, S. (2008). *Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics, 24(5), 719-720.*

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent,* http://www.stanford.edu/~hastie/Papers/glmnet.pdf

Breheny, P. and Huang, J. (2011) *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Statist., 5: 232-253.*

Examples

```
library(magrittr)

# simulation parameters
```

```
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)
betaMainInteractions <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
betaMainInteractions[which(betaMainEffect!=0)] <- runif(nActive, alphaMean - 0.1, alphaMean + 0.1)
beta <- c(betaMainEffect, betaE, betaMainInteractions)

result <- s_generate_data(p = p, X = X,
                          beta = beta,
                          include_interaction = TRUE,
                          cluster_distance = cluster_distance,
                          n = n, n0 = n0,
```

```
                          eclust_distance = Ecluster_distance,
                          signal_to_noise_ratio = SNR,
                          distance_method = distanceMethod,
                          cluster_method = clustMethod,
                          cut_method = cutMethod,
                          agglomeration_method = agglomerationMethod,
                          nPC = 1)

pen_res <- s_pen_clust(x_train = result[["X_train"]],
                       x_test = result[["X_test"]],
                       y_train = result[["Y_train"]],
                       y_test = result[["Y_test"]],
                       s0 = result[["S0"]],
                       gene_groups = result[["clustersAddon"]],
                       summary = "pc",
                       model = "lasso",
                       exp_family = "gaussian",
                       clust_type = "ECLUST",
                       include_interaction = TRUE)
unlist(pen_res)
```

`s_pen_separate` *Fit Penalized Regression Models on Simulated Data*

Description

This function can run penalized regression models on the untransformed design matrix. To be used with simulated data where the 'truth' is known i.e., you know which features are associated with the response. This function was used to produce the simulation results in Bhatnagar et al. 2016. Can run lasso, elasticnet, SCAD or MCP models

Usage

```
s_pen_separate(x_train, x_test, y_train, y_test, s0,
  exp_family = c("gaussian", "binomial"), model = c("lasso", "elasticnet",
  "scad", "mcp"), topgenes = NULL, stability = F, filter = F,
  include_E = T, include_interaction = F)
```

Arguments

`x_train`	`ntrain` x `p` matrix of simulated training set where `ntrain` is the number of training observations and `p` is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names
`x_test`	`ntest` x `p` matrix of simulated training set where `ntest` is the number of training observations and `p` is total number of predictors. This matrix needs to have named columns representing the feature names or the gene names

y_train	numeric vector of length ntrain representing the responses for the training subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class)
y_test	numeric vector of length ntest representing the responses for the test subjects. If continuous then you must set exp_family = "gaussion". For exp_family="binomial" should be either a factor with two levels, or a two-column matrix of counts or proportions (the second column is treated as the target class; for a factor, the last level in alphabetical order is the target class).
s0	chracter vector of the active feature names, i.e., the features in x_train that are truly associated with the response.
exp_family	Response type. See details for y_train argument above.
model	Regression model to be fit on cluster summaries. Default is model="lasso" which corresponds to glmnet mixing parameter alpha=1. model="elasticnet" corresponds to glmnet mixing parameter alpha=0.5, model="mcp" and model="scad" are the non-convex models from the ncvreg package
topgenes	List of features to keep if filter=TRUE. Default is topgenes = NULL which means all features are kept for the analysis
stability	Should stability measures be calculated. Default is stability=FALSE. See details
filter	Should analysis be run on a subset of features. Default is filter = FALSE
include_E	Should the environment variable be included in the regression analysis. Default is include_E = TRUE
include_interaction	Should interaction effects between the features in x_train and the environment variable be fit. Default is include_interaction=TRUE

Details

The stability of feature importance is defined as the variability of feature weights under perturbations of the training set, i.e., small modifications in the training set should not lead to considerable changes in the set of important covariates (Toloşi, L., & Lengauer, T. (2011)). A feature selection algorithm produces a weight, a ranking, and a subset of features. In the CLUST and ECLUST methods, we defined a predictor to be non-zero if its corresponding cluster representative weight was non-zero. Using 10-fold cross validation (CV), we evaluated the similarity between two features and their rankings using Pearson and Spearman correlation, respectively. For each CV fold we re-ran the models and took the average Pearson/Spearman correlation of the 10 choose 2 combinations of estimated coefficients vectors. To measure the similarity between two subsets of features we took the average of the Jaccard distance in each fold. A Jaccard distance of 1 indicates perfect agreement between two sets while no agreement will result in a distance of 0.

Value

This function has two different outputs depending on whether stability = TRUE or stability = FALSE

If `stability = TRUE` then this function returns a `p x 2` data.frame or data.table of regression coefficients without the intercept. The output of this is used for subsequent calculations of stability.

If `stability = FALSE` then returns a vector with the following elements (See Table 3: Measures of Performance in Bhatnagar et al (2016+) for definitions of each measure of performance):

`mse or AUC` Test set mean squared error if `exp_family = "gaussion"` or test set Area under the curve if `exp_family = "binomial"` calculated using the `roc` function

`RMSE` Square root of the mse. Only applicable if `exp_family = "gaussion"`

`Shat` Number of non-zero estimated regression coefficients. The non-zero estimated regression coefficients are referred to as being selected by the model

`TPR` true positive rate

`FPR` false positive rate

`Correct Sparsity` Correct true positives + correct true negative coefficients divided by the total number of features

`CorrectZeroMain` Proportion of correct true negative main effects

`CorrectZeroInter` Proportion of correct true negative interactions

`IncorrectZeroMain` Proportion of incorrect true negative main effects

`IncorrectZeroInter` Proportion of incorrect true negative interaction effects

References

Tološi, L., & Lengauer, T. (2011). *Classification with correlated features: unreliability of feature ranking and solutions. Bioinformatics, 27(14), 1986-1994.*

Bhatnagar, SR., Yang, Y., Blanchette, M., Bouchard, L., Khundrakpam, B., Evans, A., Greenwood, CMT. (2016+). *An analytic approach for interpretable predictive models in high dimensional data, in the presence of interactions with exposures* Preprint

Langfelder, P., Zhang, B., & Horvath, S. (2008). *Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. Bioinformatics, 24(5), 719-720.*

Friedman, J., Hastie, T. and Tibshirani, R. (2008) *Regularization Paths for Generalized Linear Models via Coordinate Descent,* http://www.stanford.edu/~hastie/Papers/glmnet.pdf

Breheny, P. and Huang, J. (2011) *Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. Appl. Statist., 5: 232-253.*

Examples

```
## Not run:
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
```

```
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)
betaMainInteractions <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
betaMainInteractions[which(betaMainEffect!=0)] <- runif(nActive, alphaMean - 0.1, alphaMean + 0.1)
beta <- c(betaMainEffect, betaE, betaMainInteractions)

result <- s_generate_data(p = p, X = X,
                          beta = beta,
                          include_interaction = TRUE,
                          cluster_distance = cluster_distance,
                          n = n, n0 = n0,
                          eclust_distance = Ecluster_distance,
                          signal_to_noise_ratio = SNR,
```

```
                          distance_method = distanceMethod,
                          cluster_method = clustMethod,
                          cut_method = cutMethod,
                          agglomeration_method = agglomerationMethod,
                          nPC = 1)

pen_res <- s_pen_separate(x_train = result[["X_train"]],
                          x_test = result[["X_test"]],
                          y_train = result[["Y_train"]],
                          y_test = result[["Y_test"]],
                          s0 = result[["S0"]],
                          model = "lasso",
                          exp_family = "gaussian",
                          include_interaction = TRUE)
unlist(pen_res)

## End(Not run)
```

`s_response` *Generate True Response vector for Linear Simulation*

Description

Given the true beta vector, covariates and environment variable this function generates the linear response with specified signal to noise ratio.

Usage

```
s_response(n, n0, p, genes, binary_outcome = FALSE, E,
  signal_to_noise_ratio = 1, include_interaction = FALSE, beta = NULL)
```

Arguments

`n`	Total number of subjects
`n0`	Total number of unexposed subjects
`p`	Total number of genes (or covariates)
`genes`	nxp matrix of the genes or covariates
`binary_outcome`	Logical. Should a binary outcome be generated. Default is `FALSE`. See details on how a binary outcome is generated
`E`	binary 0,1, vector of the exposure/environment variable
`signal_to_noise_ratio`	a numeric variable for the signal to noise ratio
`include_interaction`	Logical. Should the response include the interaction between E and the genes (for the non-zero `beta` coefficient vector)
`beta`	true beta coefficient vector. Assumes this vector is in the same order as the `genes`.

Value

a data.frame/data.table containing the response and the design matrix. Also an object of class `expression`

Examples

```
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
```

```
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
beta <- c(betaMainEffect, betaE)

result <- s_response(n = n, n0 = n0,
                     p = p, genes = X, binary_outcome = FALSE,
                     E = c(rep(0,n0), rep(1, n1)), signal_to_noise_ratio = 1,
                     include_interaction = FALSE,
                     beta = beta)
result[1:5,1:5]
```

`s_response_mars` *Generate True Response vector for Non-Linear Simulation*

Description

Given the covariates and environment variable this function generates the nonlinear response with specified signal to noise ratio.

Usage

```
s_response_mars(n, n0, p, genes, beta, binary_outcome = FALSE, E,
  signal_to_noise_ratio = 1, truemodule, nActive)
```

Arguments

`n`	total number of subjects
`n0`	total number of subjects with E=0
`p`	number of genes in design matrix
`genes`	nxp matrix of the genes or covariates
`beta`	true beta coefficient vector
`binary_outcome`	Logical. Should a binary outcome be generated. Default is `FALSE`. See details on how a binary outcome is generated
`E`	binary 0,1, vector of the exposure/environment variable
`signal_to_noise_ratio`	signal to noise ratio, default is 1
`truemodule`	numeric vector of the true module membership used in the `s_response_mars` function. Modules 3 and 4 are active in the response. See `s_response_mars` function for details.
`nActive`	number of active genes in the response used in the `s_response_mars`

Value

a data.frame/data.table containing the response and the design matrix. Also an object of class `expression`

Note

See Bhatnagar et al (2017+) for details on how the response is simulated.

Examples

```
library(magrittr)

# simulation parameters
rho = 0.90; p = 500 ;SNR = 1 ; n = 200; n0 = n1 = 100 ; nActive = p*0.10 ; cluster_distance = "tom";
Ecluster_distance = "difftom"; rhoOther = 0.6; betaMean = 2;
alphaMean = 1; betaE = 3; distanceMethod = "euclidean"; clustMethod = "hclust";
cutMethod = "dynamic"; agglomerationMethod = "average"

#in this simulation its blocks 3 and 4 that are important
#leaveOut:  optional specification of modules that should be left out
#of the simulation, that is their genes will be simulated as unrelated
#("grey"). This can be useful when simulating several sets, in some which a module
#is present while in others it is absent.
d0 <- s_modules(n = n0, p = p, rho = 0, exposed = FALSE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.01,
                maxCor = 1,
                corPower = 1,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE,
                leaveOut = 1:4)

d1 <- s_modules(n = n1, p = p, rho = rho, exposed = TRUE,
                modProportions = c(0.15,0.15,0.15,0.15,0.15,0.25),
                minCor = 0.4,
                maxCor = 1,
                corPower = 0.3,
                propNegativeCor = 0.3,
                backgroundNoise = 0.5,
                signed = FALSE)

truemodule1 <- d1$setLabels

X <- rbind(d0$datExpr, d1$datExpr) %>%
  magrittr::set_colnames(paste0("Gene", 1:p)) %>%
  magrittr::set_rownames(paste0("Subject",1:n))

betaMainEffect <- vector("double", length = p)

# the first nActive/2 in the 3rd block are active
betaMainEffect[which(truemodule1 %in% 3)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean - 0.1, betaMean + 0.1)

# the first nActive/2 in the 4th block are active
betaMainEffect[which(truemodule1 %in% 4)[1:(nActive/2)]] <- runif(
  nActive/2, betaMean+2 - 0.1, betaMean+2 + 0.1)
```

```
beta <- c(betaMainEffect, betaE)

result <- s_response_mars(n = n, n0 = n0,
                          p = p, genes = X, binary_outcome = TRUE,
                          E = c(rep(0,n0), rep(1, n1)), signal_to_noise_ratio = 1,
                          truemodule = truemodule1, nActive = nActive,
                          beta = beta)
result[1:5,1:5]
```

---

tcgaov *Subset of TCGA mRNA Ovarian serous cystadenocarcinoma data*

Description

A dataset containing a subset of the TCGA mRNA Ovarian serous cystadenocarcinoma data generated using Affymetrix HTHGU133a arrays. Differences in gene expression profiles have led to the identification of robust molecular subtypes of ovarian cancer; these are of biological and clinical importance because they have been shown to correlate with overall survival (Tothill et al., 2008). Improving prediction of survival time based on gene expression signatures can lead to targeted therapeutic interventions (Helland et al., 2011). The proposed ECLUST algorithm was applied to gene expression data from 511 ovarian cancer patients profiled by the Affymetrix Human Genome U133A 2.0 Array. The data were obtained from the TCGA Research Network: http://cancergenome.nih.gov/ and downloaded via the TCGA2STAT R library (Wanet al., 2015). Using the 881 signature genes from Helland et al. (2011) we grouped subjects into two groups based on the results in this paper, to create a "positive control" environmental variable expected to have a strong effect. Specifically, we defined an environment variable in our framework as: E = 0 for subtypes C1 and C2 (n = 253), and E = 1 for subtypes C4 and C5 (n = 258).

Usage

```
tcgaov
```

Format

A data.table and data.frame with 511 rows and 886 variables:

rn unique patient identifier (`character`)

subtype cancer subtype (1,2,3 or 4) as per Helland et al. 2011 (`integer`)

E binary environment variable for ECLUST method. E = 0 for subtypes 1 and 2 (n = 253), and E = 1 for subtypes 4 and 5 (n = 258) (`numeric`)

status vital status, 0 = alive, 1 = dead (`numeric`)

OS overall survival time (`numeric`)

columns 6:886 gene expression data for 881 genes. column names are the gene names (`numeric`)

Source

http://www.liuzlab.org/TCGA2STAT/#import-gene-expression

http://gdac.broadinstitute.org/

http://journals.plos.org/plosone/article/asset?unique&id=info:doi/10.1371/journal.pone.0018064.s015

References

Richard W Tothill, Anna V Tinker, Joshy George, Robert Brown, Stephen B Fox, Stephen Lade, Daryl S Johnson, Melanie K Trivett, Dariush Etemadmoghadam, Bianca Locandro, et al. Novel molecular subtypes of serous and endometrioid ovarian cancer linked to clinical outcome. Clinical Cancer Research, 14(16):5198–5208, 2008.

Aslaug Helland, Michael S Anglesio, Joshy George, Prue A Cowin, Cameron N Johnstone, Colin M House, Karen E Sheppard, Dariush Etemadmoghadam, Nataliya Melnyk, Anil K Rustgi, et al. Deregulation of mycn, lin28b and let7 in a molecular subtype of aggressive high-grade serous ovarian cancers. PloS one, 6(4):e18064, 2011.

Examples

```
# using data.table syntax from the data.table package
tcgaov[1:5, 1:10, with = FALSE]
tcgaov[,table(subtype, E, useNA = "always")]
```

---

u_cluster_similarity *Cluster similarity matrix*

Description

Return cluster membership of each predictor. This function is called internally by the s_generate_data and s_generate_data_mars functions. Is also used by the r_clust function for real data analysis.

Usage

```
u_cluster_similarity(x, expr, exprTest, distanceMethod,
  clustMethod = c("hclust", "protoclust"), cutMethod = c("dynamic", "gap",
  "fixed"), nClusters, method = c("complete", "average", "ward.D2", "single",
  "ward.D", "mcquitty", "median", "centroid"), K.max = 10, B = 50, nPC,
  minimum_cluster_size = 50)
```

Arguments

x	similarity matrix. must have non-NULL dimnames i.e., the rows and columns should be labelled, e.g. "Gene1, Gene2, ..."
expr	gene expression data (training set). rows are people, columns are genes
exprTest	gene expression test set. If using real data, and you dont have enough samples for a test set then just supply the same data supplied to the expr argument

`distanceMethod` one of "euclidean","maximum","manhattan", "canberra", "binary","minkowski" to be passed to `dist` function. If missing, then this function will take 1-x as the dissimilarity measure. This functionality is for diffCorr,diffTOM, fisherScore matrices which need to be converted to a distance type matrix.

`clustMethod` Cluster the data using hierarchical clustering or prototype clustering. Defaults `clustMethod="hclust"`. Other option is `protoclust`, however this package must be installed before proceeding with this option

`cutMethod` what method to use to cut the dendrogram. `'dynamic'` refers to `cutreeDynamicTree` library. `'gap'` is Tibshirani's gap statistic `clusGap` using the `'Tibs2001SEmax'` rule. `'fixed'` is a fixed number specified by the `nClusters` argument

`nClusters` number of clusters. Only used if `cutMethod = fixed`

`method` the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward.D", "ward.D2", "single", "complete", "average" (= UPGMA), "mcquitty" (= WPGMA), "median" (= WPGMC) or "centroid" (= UPGMC).

`K.max` the maximum number of clusters to consider, must be at least two. Only used if `cutMethod='gap'`

`B` integer, number of Monte Carlo ("bootstrap") samples. Only used if `cutMethod='gap'`

`nPC` number of principal components. Can be 1 or 2.

`minimum_cluster_size`
The minimum cluster size. Only applicable if `cutMethod='dynamic'`. This argument is passed to the `cutreeDynamic` function. Default is 50.

Value

a list of length 2:

clusters a p x 3 data.frame or data.table which give the cluster membership of each gene, where p is the number of genes. The first column is the gene name, the second column is the cluster number (numeric) and the third column is the cluster membership as a character vector of color names (these will match up exactly with the cluster number)

pcInfo a list of length 9:

eigengenes a list of the eigengenes i.e. the 1st (and 2nd if nPC=2) principal component of each module

averageExpr a data.frame of the average expression for each module for the training set

averageExprTest a data.frame of the average expression for each module for the test set

varExplained percentage of variance explained by each 1st (and 2nd if nPC=2) principal component of each module

validColors cluster membership of each gene

PC a data.frame of the 1st (and 2nd if nPC=2) PC for each module for the training set

PCTest a data.frame of the 1st (and 2nd if nPC=2) PC for each module for the test set

prcompObj the `prcomp` object

nclusters a numeric value for the total number of clusters

Examples

```
data("simdata")
X = simdata[,c(-1,-2)]
train_index <- sample(1:nrow(simdata),100)

cluster_results <- u_cluster_similarity(x = cor(X),
                                        expr = X[train_index,],
                                        exprTest = X[-train_index,],
                                        distanceMethod = "euclidean",
                                        clustMethod = "hclust",
                                        cutMethod = "dynamic",
                                        method = "average", nPC = 2,
                                        minimum_cluster_size = 75)

cluster_results$clusters[, table(module)]
names(cluster_results$pcInfo)
cluster_results$pcInfo$nclusters
```

`u_extract_selected_earth` *Get selected terms from an earth object*

Description

function to extract the selected terms from an earth object

Usage

```
u_extract_selected_earth(obj)
```

Arguments

`obj`	object of class `earth` returned by the `earth` function

Details

called internally by the `s_mars_separate` and `s_mars_clust` functions

Value

character vector of selected terms from the MARS model

u_extract_summary *Calculates cluster summaries*

Description

This is a modified version of moduleEigengenes. It can extract (1st and 2nd principal component) of modules in a given single dataset. It can also return the average, the variance explained This function is more flexible and the nPC argument is used. currently only nPC = 1 and nPC = 2 are supported

Usage

```
u_extract_summary(x_train, colors, x_test, y_train, y_test, impute = TRUE,
  nPC, excludeGrey = FALSE, grey = if (is.numeric(colors)) 0 else "grey",
  subHubs = TRUE, trapErrors = FALSE, returnValidOnly = trapErrors,
  softPower = 6, scale = TRUE, verbose = 0, indent = 0)
```

Arguments

x_train	Training data for a single set in the form of a data frame where rows are samples and columns are genes (probes, cpgs, covariates).
colors	A vector of the same length as the number of probes in expr, giving module color for all probes (genes). Color "grey" is reserved for unassigned genes.
x_test	Test set in the form of a data frame where rows are samples and columns are genes (probes, cpgs, covariates).
y_train	Training response numeric vector
y_test	Test response numeric vector
impute	If TRUE, expression data will be checked for the presence of NA entries and if the latter are present, numerical data will be imputed, using function impute.knn and probes from the same module as the missing datum. The function impute.knn uses a fixed random seed giving repeatable results.
nPC	Number of principal components and variance explained entries to be calculated. Note that only 1 or 2 is possible.
excludeGrey	Should the improper module consisting of 'grey' genes be excluded from the eigengenes?
grey	Value of colors designating the improper module. Note that if colors is a factor of numbers, the default value will be incorrect.
subHubs	Controls whether hub genes should be substituted for missing eigengenes. If TRUE, each missing eigengene (i.e., eigengene whose calculation failed and the error was trapped) will be replaced by a weighted average of the most connected hub genes in the corresponding module. If this calculation fails, or if subHubs==FALSE, the value of trapErrors will determine whether the offending module will be removed or whether the function will issue an error and stop.

`trapErrors` Controls handling of errors from that may arise when there are too many NA entries in expression data. If TRUE, errors from calling these functions will be trapped without abnormal exit. If FALSE, errors will cause the function to stop. Note, however, that subHubs takes precedence in the sense that if subHubs==TRUE and trapErrors==FALSE, an error will be issued only if both the principal component and the hubgene calculations have failed.

`returnValidOnly` logical; controls whether the returned data frame of module eigengenes contains columns corresponding only to modules whose eigengenes or hub genes could be calculated correctly (TRUE), or whether the data frame should have columns for each of the input color labels (FALSE).

`softPower` The power used in soft-thresholding the adjacency matrix. Only used when the hubgene approximation is necessary because the principal component calculation failed. It must be non-negative. The default value should only be changed if there is a clear indication that it leads to incorrect results.

`scale` logical; can be used to turn off scaling of the expression data before calculating the singular value decomposition. The scaling should only be turned off if the data has been scaled previously, in which case the function can run a bit faster. Note however that the function first imputes, then scales the expression data in each module. If the expression contain missing data, scaling outside of the function and letting the function impute missing data may lead to slightly different results than if the data is scaled within the function.

`verbose` Controls verbosity of printed progress messages. 0 means silent, up to (about) 5 the verbosity gradually increases.

`indent` A single non-negative integer controlling indentation of printed messages. 0 means no indentation, each unit above that adds two spaces.

Details

This function is called internally by the `u_cluster_similarity` function

Value

A list with the following components:

eigengenes Module eigengenes in a dataframe, with each column corresponding to one eigengene

averageExpr the average expression per module in the training set

averageExprTest the average expression per module in the training set

varExplained The variance explained by the first PC in each module

validColors A copy of the input colors with entries corresponding to invalid modules set to grey if given, otherwise 0 if colors is numeric and "grey" otherwise.

PC The 1st or 1st and 2nd PC from each module in the training set

PCTest The 1st or 1st and 2nd PC from each module in the test set

prcompObj The `prcomp` object returned by `prcomp`

nclusters the number of modules (clusters)

References

Zhang, B. and Horvath, S. (2005), "A General Framework for Weighted Gene Co-Expression Network Analysis", Statistical Applications in Genetics and Molecular Biology: Vol. 4: No. 1, Article 17

Examples

```
## Not run:
#see u_cluster_similarity for examples

## End(Not run)
```

`u_fisherZ` *Calculate Fisher's Z Transformation for Correlations*

Description

Calculate Fisher's Z transformation for correlations. This can be used as an alternative measure of similarity. Used in the `s_generate_data` function

Usage

```
u_fisherZ(n0, cor0, n1, cor1)

fisherTransform(n_1, r1, n_2, r2)
```

Arguments

`n0`	number of unexposed subjects
`cor0`	correlation matrix of unexposed covariate values. Should be dimension pxp
`n1`	number of exposed subjects
`cor1`	correlation matrix of exposed covariate values. Should be dimension pxp
`n_1`	number of unexposed subjects
`r1`	correlation for unexposed
`n_2`	number of exposed subjects
`r2`	correlation for exposed

Value

a pxp matrix of Fisher's Z transformation of correlations

Note

`fisherTransform` is called internally by `u_fisherZ` function

References

https://en.wikipedia.org/wiki/Fisher_transformation

Examples

```
data("simdata")

X = simdata[,c(-1,-2)]
fisherScore <- u_fisherZ(n0 = 100, cor0 = cor(X[1:50,]),
                         n1 = 100, cor1 = cor(X[51:100,]))

dim(fisherScore)

fisherScore[1:5,1:5]
```

Index