Package: GeneSelectR (via r-universe)

October 24, 2024

Title Comprehensive Feature Selection Worfkflow for Bulk RNAseq Datasets

Version 0.0.0.9000

Description GeneSelectR is a versatile R package designed for efficient RNA sequencing data analysis. Its key innovation lies in the seamless integration of the Python sklearn machine learning framework with R-based bioinformatics tools. This integration enables GeneSelectR to perform robust ML-driven feature selection while simultaneously leveraging the power of Gene Ontology (GO) enrichment and semantic similarity analyses. By combining these diverse methodologies, GeneSelectR offers a comprehensive workflow that optimizes both the computational aspects of ML and the biological insights afforded by advanced bioinformatics analyses. Ideal for researchers in bioinformatics, GeneSelectR stands out as a unique tool for analyzing complex RNAseq datasets with enhanced precision and relevance.

License MIT + file LICENSE

URL `https://github.com/dzhakparov/GeneSelectR`

BugReports `https://github.com/dzhakparov/GeneSelectR/issues`

Imports cowplot (>= 1.1.1), dplyr (>= 1.1.0), ggplot2 (>= 3.4.2), glue (>= 1.6.2), magrittr (>= 2.0.3), methods (>= 4.2.2), RColorBrewer (>= 1.1.3), reshape2 (>= 1.4.4), reticulate (>= 1.28), rlang (>= 1.1.1), testthat (>= 3.0.0), tibble (>= 3.2.1), tidyr (>= 1.3.0), tmod (>= 0.50.13),

Suggests clusterProfiler (>= 4.6.2), GO.db (>= 3.17.0), simplifyEnrichment (>= 1.8.0), knitr, rmarkdown, BiocManager (>= 1.30.21), UpSetR (>= 1.4.0), AnnotationHub (>= 3.8.0), ensembldb (>= 2.24.0), org.Hs.eg.db (>= 3.17.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Depends R (>= 2.10)

LazyData true

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

RemoteUrl https://github.com/dzhakparov/geneselectr

RemoteRef HEAD

RemoteSha 4e7dfd0e1538d6b38c78dd19ed57598c167445b0

Contents

`aggregate_feature_importances` *Aggregate Feature Importances*

Description

This function aggregates the feature importances for each method across all splits.

Usage

```
aggregate_feature_importances(selected_features)
```

Arguments

`selected_features`	A list of selected features. Each element of the list represents a split and should be a named list where the names are the methods and the values are data frames containing the feature importances for that method in that split.

Value

A list of aggregated feature importances for each method. Each element of the list is a data frame that contains the mean and standard deviation of the feature importances for a particular method across all splits.

Examples

```
## Not run:
  # Assuming selected_features is a list of selected features for each split
  aggregated_importances <- aggregate_feature_importances(selected_features)

## End(Not run)
```

`AnnotatedGeneLists-class` *AnnotatedGeneLists class*

Description

A class to hold a list of GeneList objects, each representing a method.

Slots

`inbuilt` A list of GeneList objects containing annotations for genes selected with inbuilt, model-specific feature importance.

`permutation` A list of GeneList objects containing annotations for genes selected with permutation importance.

`annotate_gene_lists` *Convert and Annotate Gene Lists*

Description

This function converts a list of gene lists to different formats and returns an object of the class AnnotatedGeneLists.

Usage

```
annotate_gene_lists(
  pipeline_results,
  custom_lists = NULL,
  annotations_ahb,
  format = c("ENTREZ", "ENSEMBL", "SYMBOL")
)
```

Arguments

`pipeline_results`	A PipelineResults object containing a named list of data frames.
`custom_lists`	(optional) A named list of character vectors containing additional user-defined gene sets.
`annotations_ahb`	A data.frame object containing gene annotations with columns 'gene_id', 'gene_name', and 'entrezid'.
`format`	The format of the gene list in 'pipeline_results' and 'custom_lists'. This should be one of "ENSEMBL", "ENTREZ", or "SYMBOL".

Value

An object of the class AnnotatedGeneLists.

`calculate_mean_cv_scores` *Calculate Mean Cross-Validation Scores for Various Feature Selection Methods*

Description

Calculate Mean Cross-Validation Scores for Various Feature Selection Methods

Usage

```
calculate_mean_cv_scores(selected_pipelines, cv_best_score)
```

Arguments

`selected_pipelines`	A list of pipelines for different feature selection methods.
`cv_best_score`	A list or vector of cross-validation scores.

Value

A dataframe containing the mean and standard deviation of cross-validation scores for each method.

`calculate_overlap_coefficients` *Calculate Overlap and Similarity Coefficients between Feature Lists*

Description

This function calculates the Overlap, Jaccard, and Soerensen-Dice coefficients to quantify the similarity between feature lists. In addition to feature importance and permutation importance, you can provide a custom list of feature names to be included in the overlap calculation.

Usage

```
calculate_overlap_coefficients(pipeline_results, custom_lists = NULL)
```

Arguments

`pipeline_results`	A PipelineResults object containing the fitted pipelines, cross-validation results, selected features, mean performance, and mean feature importances.
`custom_lists`	An optional named list of character vectors. Each character vector should contain feature names. The names of the list will be used as names in the resulting overlap coefficient matrices.

Value

A list of lists of matrices showing the Overlap, Jaccard, and Soerensen-Dice coefficients for the feature lists. This includes coefficients for feature importance, permutation importance (if present), and custom lists (if provided).

`calculate_permutation_feature_importance` *Calculate Permutation Feature Importance*

Description

This function calculates permutation feature importance for a Scikit-learn pipeline with a trained classifier as the final step.

Usage

```
calculate_permutation_feature_importance(
  pipeline,
  X_train,
  y_train,
  n_repeats = 10L,
  random_state = 0L,
  njobs = njobs,
  pipeline_name,
  iter
)
```

Arguments

`pipeline`	A Scikit-learn pipeline object with a trained classifier as the final step.
`X_train`	A DataFrame containing the training data.
`y_train`	A DataFrame containing the training labels.
`n_repeats`	An integer specifying the number of times to permute each feature.
`random_state`	An integer specifying the seed for the random number generator.
`njobs`	An integer specifying number of cores to use. Set up by the master GeneSelectR function.
`pipeline_name`	Strings (names of the selected_pipelines list) representing pipeline names that were constructed for the feature selection
`iter`	An integer that is indicating current iteration of the train-test split

Value

A list containing the feature names and their importance scores.

Examples

```
## Not run:
# Assuming you have a Scikit-learn pipeline 'my_pipeline' and training data 'X_train'
permutation_importances <- calculate_permutation_feature_importance(my_pipeline, X_train, y_train)

## End(Not run)
```

`compute_GO_child_term_metrics`
Retrieve and Plot the Offspring Nodes of GO Terms

Description

This function retrieves the children nodes for a set of Gene Ontology (GO) terms from a list of GO terms and can plot the offspring nodes' numbers and fractions for each term.

Usage

```
compute_GO_child_term_metrics(GO_data, GO_terms, ontology = "BP", plot = FALSE)
```

Arguments

`GO_data`	A list of GO data where each element corresponds to a different feature list. Each element should have a @result data frame with a column 'ID' containing GO terms.
`GO_terms`	A character vector containing GO term IDs for which offspring nodes are to be fetched.
`ontology`	A character string specifying the type of ontology to be considered. Can be one of 'BP' (Biological Process), 'MF' (Molecular Function), or 'CC' (Cellular Component). Default is 'BP'.
`plot`	A logical. If TRUE, the function plots the number and fraction of offspring nodes for each term in `GO_terms` across all feature lists in `GO_data`. Default is FALSE.

Value

A data frame with columns:

- **feature_list** - The names of the feature lists from `GO_data`.
- **all_terms_number** - The total number of GO terms in each feature list.
- **offspring_nodes_number** - The number of offspring nodes for the given GO term(s) in each feature list.
- **offspring_terms** - The actual offspring terms for the given GO term(s) in each feature list, concatenated by ';'.
- **fraction** - The fraction (percentage) of offspring nodes out of all terms in each feature list.
- **GO_term** - The GO term being considered.

Examples

```
## Not run:
GO_terms_vec <- c("GO:0002376", "GO:0008150")
df_res <- get_children_nodes(GO_data = all_selection.GO_inbuilt,
                             GO_terms = GO_terms_vec,
                             plot = TRUE)

## End(Not run)
```

`configure_environment` *Configure Python Environment for GeneSelectR*

Description

This function checks if Conda is installed, creates a new Conda environment (if it does not already exist), installs necessary Python packages into the environment, and sets it as the active environment for reticulate.

Usage

```
configure_environment(env_name = "GeneSelectR_env")
```

Arguments

`env_name`	The name of the Conda environment to be created. Defaults to "GeneSelectR_env".

Value

A message indicating the status of the environment configuration. If the configuration is successful, the function will prompt the user to restart their R session for the changes to take effect.

Examples

```
## Not run:
# Configure the default environment
configure_environment()

# Configure a custom environment
configure_environment("my_env_name")

## End(Not run)
```

`create_conda_env` *Create a specific Conda environment*

Description

This function creates a Conda environment if it doesn't already exist.

Usage

```
create_conda_env(conda_env = "GeneSelectR_env")
```

Arguments

`conda_env`	The name of the Conda environment to create.

`create_pipelines` *Create Pipelines*

Description

This function creates a list of Scikit-learn pipelines using the specified feature selection methods, preprocessing steps, and classifier.

Usage

```
create_pipelines(
  feature_selection_methods,
  preprocessing_steps,
  selected_methods,
  classifier,
  fs_param_grids
)
```

Arguments

`feature_selection_methods`	A list of feature selection methods to use for the pipelines.
`preprocessing_steps`	A list of preprocessing steps to use for the pipelines.
`selected_methods`	A vector of names of feature selection methods to use from the default set.
`classifier`	A Scikit-learn classifier to use as the final step in the pipelines.
`fs_param_grids`	param grid

Value

A list of Scikit-learn pipelines.

`create_test_metrics_df` *Create a Dataframe of Test Metrics*

Description

Create a Dataframe of Test Metrics

Usage

```
create_test_metrics_df(test_metrics)
```

Arguments

`test_metrics`	A list or dataframe of test metrics.

Value

A dataframe with the processed test metrics.

`define_sklearn_modules` *Define Python modules and scikit-learn submodules*

Description

Define Python modules and scikit-learn submodules

Usage

```
define_sklearn_modules()
```

Value

A list containing the definitions for the Python modules and submodules.

`enable_multiprocess` *Enable Multiprocessing in Python Environment*

Description

This function sets up the necessary executable for Python's multiprocessing functionality. Only used on Windows

Usage

```
enable_multiprocess()
```

`evaluate_test_metrics` *Evaluate Test Metrics for a Grid Search Model*

Description

This function takes a grid search object, test data, and test labels to evaluate the performance of the best model found during grid search.

Usage

```
evaluate_test_metrics(grid_search, X_test, y_test, modules)
```

Arguments

`grid_search`	A grid search object containing the best estimator.
`X_test`	A data frame or matrix of test features.
`y_test`	A vector of test labels.
`modules`	A list of Python modules used in the function.

Value

A list containing the weighted precision, weighted recall, weighted F1 score, and accuracy.

Examples

```
## Not run:
# Assuming grid_search, X_test, y_test, and sklearn are defined
metrics <- evaluate_test_metrics(grid_search, X_test, y_test, sklearn)

## End(Not run)
```

`GeneList-class` *GeneList class*

Description

A class to hold annotated gene list for a single method.

Slots

`SYMBOL` A character vector of gene names.

`ENSEMBL` A character vector of Ensembl IDs.

`ENTREZID` A character vector of Entrez IDs.

GeneSelectR *Gene Selection and Evaluation with GeneSelectR*

Description

This function performs gene selection using different methods on a given training set and evaluates their performance using cross-validation. Optionally, it also calculates permutation feature importances.

Usage

```
GeneSelectR(
  X,
  y,
  pipelines = NULL,
  custom_fs_methods = NULL,
  selected_methods = NULL,
  custom_fs_grids = NULL,
  classifier = NULL,
  classifier_grid = NULL,
  preprocessing_steps = NULL,
  testsize = 0.2,
  validsize = 0.2,
  scoring = "accuracy",
  njobs = -1,
  n_splits = 5,
  search_type = "random",
  n_iter = 10,
  max_features = 50,
  calculate_permutation_importance = FALSE,
  perform_test_split = FALSE,
  random_state = NULL
)
```

Arguments

`X`	A matrix or data frame with features as columns and observations as rows.
`y`	A vector of labels corresponding to the rows of X_train.
`pipelines`	An optional list of pre-defined pipelines to use for fitting and evaluation. If this argument is provided, the feature selection methods and preprocessing steps will be ignored.
`custom_fs_methods`	An optional list of feature selection methods to use for fitting and evaluation. If this argument is not provided, a default set of feature selection methods will be used.

`selected_methods`	An optional vector of names of feature selection methods to use from the default set. If this argument is provided, only the specified methods will be used.
`custom_fs_grids`	An optional list of hyperparameter grids for the feature selection methods. Each element of the list should be a named list of parameters for a specific feature selection method. The names of the elements should match the names of the feature selection methods. If this argument is provided, the function will perform hyperparameter tuning for the specified feature selection methods in addition to the final estimator.
`classifier`	An optional sklearn classifier. If left NULL then sklearn RandomForestClassifier is used.
`classifier_grid`	An optional named list of classifier parameters. If none are provided then default grid is used (check vignette for exact params).
`preprocessing_steps`	An optional named list of sklearn preprocessing procedures. If none provided defaults are used (check vignette for exact params).
`testsize`	The size of the test set used in the evaluation.
`validsize`	The size of the validation set used in the evaluation.
`scoring`	A string representing what scoring metric to use for hyperparameter adjustment. Default value is 'accuracy'
`njobs`	Number of jobs to run in parallel.
`n_splits`	Number of train/test splits.
`search_type`	A string indicating the type of search to use. 'grid' for GridSearchCV and 'random' for RandomizedSearchCV. Default is 'random'.
`n_iter`	An integer indicating the number of parameter settings that are sampled in RandomizedSearchCV. Only applies when search_type is 'random'.
`max_features`	Maximum number of features to be selected by default feature selection methods.
`calculate_permutation_importance`	A boolean indicating whether to calculate permutation feature importance. Default is FALSE.
`perform_test_split`	Whether to perform train and test split, to have an evaluation on unseen test set. The default value is set to FALSE
`random_state`	An integer value setting the random seed for feature selection algorithms and cross validation procedure. By default set to NULL to use different random seed every time an algorithm is used. For reproducibility could be fixed, otherwise for an unbiased estimation should be left as NULL.

Value

A list with the following elements:

`fitted_pipelines`	A list of the fitted pipelines.
`cv_results`	A list of the cross-validation results for each pipeline.
`inbuilt_feature_importance`	A list of the inbuilt feature importance scores for each pipeline.
`gene_set_stability`	A list of the gene set stability for each pipeline.
`test_metrics`	A data frame of test metrics for each pipeline.
`permutation_importances`	A list of the permutation importance scores for each pipeline (if calculate_permutation_importance is TRUE).

Examples

```
## Not run:
# Perform gene selection and evaluation using the default methods
data(iris)
X <- iris[,1:4]
y <- iris[,5]
results <- GeneSelectR(X_train = X, y_train = y)

# Perform gene selection and evaluation using a subset of the default methods
results <- GeneSelectR(X_train = X, y_train = y, selected_methods = c("Univariate", "RFE"))

# Perform gene selection and evaluation using user-defined methods
fs_methods <- list("Lasso" = select_model(lasso(penalty = 'l1',
                                               C = 0.1,
                                               solver = 'saga'),
                                         threshold = 'median'))
custom_fs_grids <- list("Lasso" = list('C' = c(0.1, 1, 10)))
results <- GeneSelectR(X_train = X,
                       y_train = y,
                       custom_fs_methods = fs_methods,
                       custom_fs_grids = custom_fs_grids)

## End(Not run)
```

`get_feature_importances` *Get Feature Importances*

Description

This function extracts feature importances from a Scikit-learn pipeline that has a Gradient Boosting Classifier as the final step.

Usage

```
get_feature_importances(pipeline, X_train, pipeline_name, iter)
```

Arguments

`pipeline`	A Scikit-learn pipeline object with a Gradient Boosting Classifier as the final step.
`X_train`	A DataFrame containing the training data.
`pipeline_name`	Strings (names of the selected_pipelines list) representing pipeline names that were constructed for the feature selection
`iter`	An integer that is indicating current iteration of the train-test split

Value

A list containing the selected feature names and their importances, or NULL if the classifier is not a Gradient Boosting Classifier or the feature selector doesn't have the 'get_support' method.

Examples

```
## Not run:
# Assuming you have a Scikit-learn pipeline 'my_pipeline' and training data 'X_train'
feature_importances <- get_feature_importances(my_pipeline, X_train)
# Extract the selected feature names and their importances
selected_features <- feature_importances$selected_features
importances <- feature_importances$importances

## End(Not run)
```

`GO_enrichment_analysis`
Perform gene set enrichment analysis using clusterProfiler

Description

This function performs gene set enrichment analysis on a list of gene sets extracted from an AnnotatedGeneLists object.

Usage

```
GO_enrichment_analysis(
  annotated_gene_lists,
  list_type = "inbuilt",
  background = NULL,
  organism = "org.Hs.eg.db",
  keyType = "ENTREZID",
  ont = "BP",
  pvalueCutoff = 0.05,
  qvalueCutoff = 0.2,
  pAdjMethod = "fdr",
  ...
)
```

Arguments

`annotated_gene_lists`	An AnnotatedGeneLists object containing a list of GeneList objects.
`list_type`	A type of AnnotatedGeneList from annotate_gene_lists function. Either 'inbuilt' or 'permutation'. (default: 'inbuilt')
`background`	A character vector representing the background gene set.
`organism`	A character string corresponding to the organism of interest. Default: "org.Hs.eg.db" (for human).
`keyType`	A character string indicating the type of gene identifiers. Default: "ENTREZID".
`ont`	A character string representing GO term ontology. Default: "BP" (for Biological Process).
`pvalueCutoff`	A numeric value specifying the significance cutoff. Default: 0.05.
`qvalueCutoff`	A numeric value specifying the q-value cutoff. Default: 0.2.
`pAdjMethod`	A p-value adjustment method. Should be the one from "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". Default is 'fdr
`...`	Other parameters to be passed to clusterProfiler::enrichGO function.

Value

A list containing the enrichment results for all gene sets (excluding "background").

References

To use clusterProfiler in published research, please cite: Yu G, Wang LG, Han Y, He QY. clusterProfiler: an R package for comparing biological themes among gene clusters. OMICS: A Journal of Integrative Biology. 2012;16(5):284-287. doi:10.1089/omi.2011.0118.

Examples

```
## Not run:
# Assuming annotated_gene_lists and background are defined
results <- GO_enrichment_analysis(annotated_gene_lists,
                                  background,
                                  organism = "org.Hs.eg.db",
                                  keyType = "ENTREZID")

## End(Not run)
```

`import_python_packages`
Import Python Libraries

Description

This function imports the necessary Python libraries for the package.

Usage

```
import_python_packages()
```

Value

A list of imported Python libraries.

`install_python_packages`
Install necessary Python packages in a specific Conda environment

---

Description

This function installs the necessary Python packages in a specific Conda environment.

Usage

```
install_python_packages(conda_env = "GeneSelectR_env")
```

Arguments

`conda_env`	The name of the Conda environment to use.

perform_grid_search *Perform Grid Search or Random Search for Hyperparameter Tuning*

Description

Perform Grid Search or Random Search for Hyperparameter Tuning

Usage

```
perform_grid_search(
  X_train,
  y_train,
  pipeline,
  scoring,
  params,
  search_type,
  n_iter,
  njobs,
  modules,
  random_state
)
```

Arguments

`X_train`	Training data for predictors.
`y_train`	Training data for outcomes.
`pipeline`	A pipeline specifying the steps for feature selection and model training.
`scoring`	A string representing what scoring metric to use for hyperparameter adjustment. Default value is 'accuracy'
`params`	A list of parameters or parameter distributions to search over.
`search_type`	A character string specifying the type of search ('grid' or 'random').
`n_iter`	The number of parameter settings that are sampled in a random search.
`njobs`	The number of CPU cores to use.
`modules`	A list containing the definitions for the Python modules and submodules.
`random_state`	An integer value setting the random seed for feature selection algorithms and cross validation procedure. By default set to NULL to use different random seed every time an algorithm is used. For reproducibility could be fixed, otherwise for an unbiased estimation should be left as NULL.

Value

An object of the optimal model found during the search.

`PipelineResults-class` *PipelineResults class*

Description

A class to hold the results of GeneSelectR function.

Slots

`best_pipeline` A named list containing parameters of the best performer pipeline

`cv_results` A list of the cross-validation results for each pipeline.

`inbuilt_feature_importance` A list of the inbuilt mean feature importances for each method across all splits.

`permutation_importance` A list of the permutation importances for each method.

`cv_mean_score` A data.frame of mean scores from cross-validation.

`test_metrics` A data.frame containing metrics (F1, accuracy, precision, and recall) calculated on the unseen test set. Contains mean values across splits as well as standard deviation.

`pipeline_to_list` *Convert Scikit-learn Pipeline to Named List*

Description

This function takes a Scikit-learn Pipeline object and converts it to a named list in R. Each step in the pipeline becomes an element in the list with the name of the step as the name of the list element.

Usage

```
pipeline_to_list(pipeline)
```

Arguments

`pipeline`	A Scikit-learn Pipeline object.

Value

A named list where each element represents a step in the pipeline. The names of the list elements correspond to the names of the steps in the pipeline.

`plot_feature_importance` *Plot Feature Importance*

Description

This function plots the feature importance scores from `inbuilt_feature_importance` and `permutation_importance` in the `PipelineResults` object.

Usage

```
plot_feature_importance(pipelineresults, top_n_features = 10)
```

Arguments

`pipelineresults`	An object of class `PipelineResults`.
`top_n_features`	An integer specifying the top N features to plot based on their mean importance.

Value

A list of grid of ggplot objects.

`plot_metrics` *Plot Performance Metrics*

Description

This function creates separate plots for f1_mean, recall_mean, precision_mean, and accuracy_mean from a given dataframe, then arranges these plots using cowplot::plot_grid. Requires ggplot2 and cowplot packages.

Usage

```
plot_metrics(pipeline_results)
```

Arguments

`pipeline_results`	An object of class "PipelineResults" containing the performance metrics. This object is expected to contain a dataframe with the columns 'method', 'f1_mean', 'f1_sd', 'recall_mean', 'recall_sd', 'precision_mean', 'precision_sd', 'accuracy_mean', 'accuracy_sd'.

Value

A combined ggplot object. If no test metrics provided, only CV performance plot is returned

plot_overlap_heatmaps *Generate Heatmaps to Visualize Overlap and Similarity Coefficients between Feature Lists*

Description

This function takes a list of matrices of overlap and similarity coefficients and generates heatmaps to visualize them.

Usage

```
plot_overlap_heatmaps(coefficients, save_plot = FALSE, filename = NULL)
```

Arguments

`coefficients`	A list of matrices showing the Overlap, Jaccard, and Soerensen-Dice coefficients for the feature lists.
`save_plot`	A logical value indicating whether to save the heatmap plots to a file or not. Default is FALSE.
`filename`	A character string specifying the filename for the saved heatmap plots (if save_plot = TRUE).

Value

A grid of heatmaps showing the Overlap, Jaccard, and Soerensen-Dice coefficients for the feature lists.

plot_upset *Plot Feature Overlaps Using UpSet Plots*

Description

This function produces separate UpSet plots for inbuilt feature importances and permutation importances, allowing you to visualize the overlap of feature lists. Optionally, you can include custom lists.

Usage

```
plot_upset(pipeline_results, custom_lists = NULL)
```

Arguments

`pipeline_results`	A PipelineResults object containing the fitted pipelines, cross-validation results, selected features, mean performance, and mean feature importances.
`custom_lists`	An optional named list of character vectors. Each character vector should contain feature names. The names of the list will be used as names in the UpSet plots.

Value

A named list containing two UpSet plots accessible as `result$inbuilt_importance` for inbuilt feature importances and `result$permutation_importance` for permutation importances.

Examples

```
## Not run:
pipeline_results <- PipelineResults$new(...)
custom_lists <- list("custom1" = c("gene1", "gene2"), "custom2" = c("gene3", "gene4"))
result <- plot_feature_overlap_upset(pipeline_results, custom_lists)
print(result$plot1)
print(result$plot2)

## End(Not run)
```

`run_multiple_simplifyGO` *Run multiple simplifyGOFromMultipleLists*

Description

This function runs the simplifyGOFromMultipleLists function with different combinations of semantic similarity measures and clustering methods.

Usage

```
run_multiple_simplifyGO(all_selection_GO, padj_column, padj_cutoff, ont)
```

Arguments

`all_selection_GO`	A list of dataframes. The GO object for the simplifyGOFromMultipleLists function.
`padj_column`	Character. The column name for the p-value adjustment.
`padj_cutoff`	Numeric. The cutoff for the p-value adjustment.
`ont`	Character. The ontology for the simplifyGOFromMultipleLists function.

Value

A named list of heatmaps generated by the simplifyGOFromMultipleLists function.

run_simplify_enrichment
Run simplifyGOFromMultipleLists with specified measure and method

Description

This function is simply a wrapper for the simplifyGOFromMultipleLists function in the simplifyEnrichment package, created for the ease of data input. All credit for the underlying functionality goes to the authors of the simplifyEnrichment package.

Usage

```
run_simplify_enrichment(
  fs_GO_results,
  padj_column = "p.adjust",
  padj_cutoff,
  ont,
  measure,
  method,
  ...
)
```

Arguments

`fs_GO_results`	A list of dataframes containing GO enrichment results for feature selection methods. The GO object for the simplifyGOFromMultipleLists function.
`padj_column`	Character. The column name for the p-value adjustment.
`padj_cutoff`	Numeric. The cutoff for the p-value adjustment.
`ont`	Character. The ontology for the simplifyGOFromMultipleLists function.
`measure`	Character. The semantic similarity measure for the simplifyGOFromMultipleLists function.
`method`	Character. The clustering method for the simplifyGOFromMultipleLists function.
`...`	Other parameters that can be passed to simplifyGOFromMultipleLists

Value

The result of the simplifyGOFromMultipleLists function.

set_default_fs_methods *Set Default Feature Selection Methods*

Description

Set Default Feature Selection Methods

Usage

```
set_default_fs_methods(modules, max_features, random_state)
```

Arguments

modules	A list containing the definitions for the Python modules and submodules.
max_features	The maximum number of features to consider.
random_state	An integer value setting the random seed for feature selection algorithms and cross validation procedure. By default set to NULL to use different random seed every time an algorithm is used. For reproducibility could be fixed, otherwise for an unbiased estimation should be left as NULL.

Value

A list containing preprocessing steps and default feature selection methods.

set_default_param_grids *Set Default Parameter Grids for Feature Selection*

Description

Set Default Parameter Grids for Feature Selection

Usage

```
set_default_param_grids(max_features)
```

Arguments

max_features	An integer indicating max_features to select in Univariate select

Value

A list containing the default parameter grids for feature selection methods.

`set_reticulate_python` *Set RETICULATE_PYTHON for the Current Session*

Description

This function sets the RETICULATE_PYTHON environment variable to the path of the Python interpreter in the specified Conda environment, but only for the current R session. The change will not persist after the R session is closed.

Usage

```
set_reticulate_python(env_name = "GeneSelectR_env")
```

Arguments

`env_name`	The name of the Conda environment. Default is 'GeneSelectR_env'.

Details

This function checks if the specified Conda environment exists. If it does, the function sets the RETICULATE_PYTHON environment variable to the path of the Python interpreter in that environment. If the environment does not exist, the function stops with an error message.

Users need to run this function in every new R session where they want to use your package. Also, they should run this function before loading your package with library(), because the RETICULATE_PYTHON environment variable needs to be set before reticulate is loaded.

Value

This function does not return a value. It sets the RETICULATE_PYTHON environment variable for the current R session and prints a message indicating the new value of RETICULATE_PYTHON.

Examples

```
## Not run:
set_reticulate_python('GeneSelectR_env')

## End(Not run)
```

`skip_if_no_modules` *Check if Python Modules are Available*

Description

This helper function checks if a list of Python modules are available. If any are not, it skips the tests.

Usage

```
skip_if_no_modules(module_names)
```

Arguments

`module_names`	A vector of names of the Python modules to check.

`split_data` *Split Data into Training and Test Sets*

Description

Split Data into Training and Test Sets

Usage

```
split_data(X, y, test_size, modules)
```

Arguments

`X`	A dataframe or matrix of predictors.
`y`	A vector of outcomes.
`test_size`	Proportion of the data to be used as the test set.
`modules`	A list containing the definitions for the Python modules and submodules.

Value

A list containing the training and test sets for predictors and outcomes.

`steps_to_tuples` *Convert Steps to Tuples*

Description

This function converts a list of steps to tuples for use in a Scikit-learn pipeline.

Usage

```
steps_to_tuples(steps)
```

Arguments

`steps`	A list of steps to convert to tuples.

Value

A list of tuples.

Index