Package: MOFSR (via r-universe)

June 6, 2026

Type Package

Title Multi-Omics Fusion for Subtype Recognition

Version 2.3.0

Author Zaoqu Liu [aut, cre] (<https://orcid.org/0000-0002-0452-742X>)

Maintainer Zaoqu Liu <liuzaoqu@163.com>

Description A comprehensive toolkit for integrating multi-modal biological data to discover disease subtypes and biological mechanisms. MOFSR provides 15 state-of-the-art multi-omics clustering algorithms (SNF, wSNF, CPCA, iClusterBayes, IntNMF, LRAcluster, MCIA, MOFA, NEMO, PINSPlus, RGCCA, SGCCA, CIMLR, BCC, LateFusion), 17 classification methods, parallel computing support, comprehensive visualization (UMAP, heatmaps, survival curves), data preprocessing (normalization, filtering, batch correction, QC), feature selection with bootstrap validation, and cluster quality assessment. All core algorithms are implemented internally for maximum compatibility, cross-platform support, and optimized performance. Works on Windows, macOS, and Linux without external dependencies for core functionality.

License GPL (>= 3)

URL https://zaoqu-liu.github.io/MOFSR/, https://github.com/Zaoqu-Liu/MOFSR

BugReports https://github.com/Zaoqu-Liu/MOFSR/issues

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports stats, graphics

Suggests testthat (>= 3.0.0), knitr, rmarkdown, bench, ggplot2, glmnet, nnet, e1071, randomForest, xgboost, survival, survminer, dplyr, tibble, GSVA, BiocManager, BiocGenerics, BiocParallel, Hmisc, MASS, adabag, caret, furrr, future,

future.apply, gbm, mlr3, mlr3learners, progress, purrr, rpart, stringr, class, parallel

RoxygenNote 7.3.3

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://zaoqu-liu.r-universe.dev

Date/Publication 2026-01-29 07:35:37 UTC

RemoteUrl https://github.com/Zaoqu-Liu/MOFSR

RemoteRef master

RemoteSha ebcefdc51eb166ecacbe12441132af1af78d3457

Contents

algo-factor . . . 5
algo-iclusterbayes . . . 6
algo-nemo . . . 7
algo-snf . . . 7
align_samples . . . 8
bcc_cluster . . . 8
bcc_cluster_fast . . . 9
calc_chi . . . 10
calc_pac . . . 10
CalCHI . . . 11
CalPAC . . . 11
check_sample_alignment . . . 12
cimlr_cluster . . . 13
cimlr_feature_ranking . . . 13
Classifier.Adaboost . . . 14
Classifier.DT . . . 15
Classifier.Enet . . . 17
Classifier.Enrichment . . . 19
Classifier.GBDT . . . 20
Classifier.kNN . . . 22
Classifier.LASSO . . . 24
Classifier.LDA . . . 25
Classifier.NBayes . . . 27
Classifier.NNet . . . 29
Classifier.PCA . . . 30
Classifier.RF . . . 32
Classifier.Ridge . . . 34
Classifier.ssGSEA . . . 35
Classifier.StepLR . . . 37
Classifier.SVD . . . 39
Classifier.SVM . . . 41
Classifier.XGBoost . . . 42

compare_clusterings . . . 44
compute_umap . . . 44
consensus_cluster . . . 45
correct_batch . . . 46
cpca . . . 46
CV . . . 47
CV.df . . . 47
FeatureSelectionWithBootstrap . . . 48
filter_by_mad . . . 49
filter_low_variance . . . 49
Find.OptClusterFeatures . . . 50
gene_sets . . . 51
get.binary.clusters . . . 51
get.class . . . 52
get.Jaccard.Distance . . . 52
get_consensus_class . . . 53
handle_missing . . . 54
icluster_bayes_fast . . . 54
init . . . 55
intnmf_cluster . . . 55
intnmf_opt_k . . . 56
lracluster . . . 57
MAD.df . . . 57
mcia . . . 58
Mean.df . . . 58
Median.df . . . 59
minmax . . . 59
minmax.df . . . 60
nemo_affinity_graph . . . 60
nemo_clustering . . . 61
nemo_num_clusters . . . 61
parallel . . . 62
parallel_bootstrap_features . . . 62
parallel_consensus_cluster . . . 63
PathDEA . . . 64
perturbation_clustering . . . 65
plot_algorithm_comparison . . . 66
plot_cluster_quality . . . 67
plot_consensus_heatmap . . . 67
plot_silhouette . . . 68
plot_survival . . . 69
plot_umap . . . 69
preprocessing . . . 70
qc_summary . . . 71
rgcca . . . 71
run-clustering . . . 72
run_bcc . . . 73
run_cimlr . . . 73

run_cpca . . . 74
run_iclusterbayes . . . 74
run_integration . . . 75
run_intnmf . . . 76
run_late_fusion . . . 76
run_lracluster . . . 77
run_mcia . . . 77
run_mofa . . . 78
run_multiple_algorithms . . . 78
run_nemo . . . 79
run_parallel_algorithms . . . 79
run_pinsplus . . . 80
run_rgcca . . . 80
run_sgcca . . . 81
run_snf . . . 82
run_wsnf . . . 82
RunBCC . . . 83
RunCC . . . 84
RunCIMLR . . . 85
RunClassifier . . . 85
RunCOCA . . . 87
RunCPCA . . . 88
RunEnsemble . . . 88
RunGSVA . . . 90
RuniClusterBayes . . . 92
RunIntegration . . . 93
RunIntNMF . . . 94
RunLRAcluster . . . 95
RunMCIA . . . 96
RunMOFS . . . 96
RunNEMO . . . 98
RunPCA . . . 99
RunPINSPlus . . . 100
RunRGCCA . . . 100
RunSGCCA . . . 101
RunSNF . . . 102
SD.df . . . 103
Select.Features . . . 104
sgcca . . . 105
snf_fuse . . . 106
spectral_clustering . . . 106
ssMwwGST . . . 107
stop_parallel . . . 107
subtyping_omics_data . . . 108
WangGBM . . . 108
WuGBM . . . 110

Index **113**

algo-factor *Multi-View Factor Analysis*

Description

Factor analysis methods for multi-omics data integration.

Performs multi-view factor analysis for multi-omics integration. This is a simplified Bayesian factor model with ARD priors for sparsity. For the full MOFA implementation, use the MOFA2 Bioconductor package.

Usage

```
multi_view_factor_analysis(
  data_list,
  n_factors = 10,
  max_iter = 1000,
  tol = 1e-05,
  sparsity_prior = TRUE,
  verbose = FALSE
)
```

Arguments

`data_list`	List of data matrices (features x samples).
`n_factors`	Number of latent factors (default: 10).
`max_iter`	Maximum iterations (default: 1000).
`tol`	Convergence tolerance (default: 1e-5).
`sparsity_prior`	Use ARD prior for sparsity (default: TRUE).
`verbose`	Print progress (default: FALSE).

Details

This function implements a simplified multi-view factor model: Y_m = W_m * Z^T + E_m, where Y_m is the data for view m, W_m are the loadings, Z are the latent factors shared across views, and E_m is Gaussian noise. ARD (Automatic Relevance Determination) priors are used for automatic factor pruning.

Value

List with factors (Z), weights (W), variance explained, and convergence info.

Note

For production use of MOFA, we recommend the MOFA2 Bioconductor package which provides the full implementation with Python backend.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

`algo-iclusterbayes` *Bayesian Integrative Clustering (iClusterBayes)*

Description

Pure R implementation of iClusterBayes for multi-omics clustering.

Performs Bayesian integrative clustering using Gibbs sampling.

Usage

```
icluster_bayes(
  data_list,
  k,
  n_burnin = 1000,
  n_sample = 2000,
  n_thin = 1,
  prior_alpha = 1,
  prior_sigma = c(1, 1),
  verbose = FALSE
)
```

Arguments

`data_list`	List of data matrices (features x samples).
`k`	Number of clusters.
`n_burnin`	Number of burn-in iterations (default: 1000).
`n_sample`	Number of sampling iterations (default: 2000).
`n_thin`	Thinning interval (default: 1).
`prior_alpha`	Dirichlet prior parameter (default: 1).
`prior_sigma`	Prior for residual variance (default: c(1, 1)).
`verbose`	Print progress.

Value

List with cluster assignments, posterior matrices, and parameters.

Author(s)

Zaoqu Liu

References

Mo Q, et al. A fully Bayesian latent variable model for integrative clustering analysis of multi-type omics data. Biostatistics. 2018.

algo-nemo *Neighborhood-based Multi-Omics clustering (NEMO)*

Description

Internal implementation of NEMO algorithm for multi-omics clustering.

Usage

```
.NEMO_NEIGHBORS_RATIO
```

Format

An object of class `numeric` of length 1.

Author(s)

Zaoqu Liu

References

Rappoport N, Shamir R. NEMO: cancer subtyping by integration of partial multi-omic data. Bioinformatics. 2019.

algo-snf *Similarity Network Fusion (SNF) Algorithm*

Description

Internal implementation of SNF algorithm for multi-omics data integration.

Computes an affinity matrix using local Gaussian kernel.

Usage

```
snf_affinity_matrix(diff, K = 20, sigma = 0.5)
```

Arguments

`diff`	Distance matrix (squared Euclidean distances).
`K`	Number of nearest neighbors for local scaling (default: 20).
`sigma`	Variance for local model (default: 0.5).

Value

Affinity matrix with exponential similarity.

Author(s)

Zaoqu Liu

References

Wang B, et al. Similarity Network Fusion for Aggregating Data Types on a Genomic Scale. Nat Methods. 2014;11(3):333-337.

Wang B, et al. Nat Methods. 2014 - Equation in Methods section

---

`align_samples` *Align Samples Across Datasets*

Description

Aligns samples across multiple datasets to common samples.

Usage

```
align_samples(data_list)
```

Arguments

`data_list`	List of data matrices.

Value

List of aligned matrices with common samples.

`bcc_cluster` *BCC Core Algorithm*

Description

Performs Bayesian Consensus Clustering.

Usage

```
bcc_cluster(
  data_list,
  k,
  n_iter = 1000,
  n_burnin = 500,
  alpha = 1,
  verbose = FALSE
)
```

Arguments

data_list	List of data matrices (features x samples).
k	Number of clusters.
n_iter	Number of MCMC iterations (default: 1000).
n_burnin	Number of burn-in iterations (default: 500).
alpha	Dirichlet prior concentration (default: 1).
verbose	Print progress.

Value

List with cluster assignments and consensus matrix.

---

bcc_cluster_fast *Fast BCC*

Description

Faster version of BCC using EM-like approach.

Usage

```
bcc_cluster_fast(data_list, k, max_iter = 50)
```

Arguments

data_list	List of data matrices.
k	Number of clusters.
max_iter	Maximum iterations (default: 50).

Value

List with cluster assignments.

calc_chi *Calculate Calinski-Harabasz Index*

Description

Computes CH index to evaluate clustering quality.

Usage

```
calc_chi(
  hclust_result,
  dist_matrix = NULL,
  max_clusters = round(1 + 3.3 * log10(length(hclust_result$order)))
)
```

Arguments

hclust_result	Hierarchical clustering result.
dist_matrix	Optional distance matrix.
max_clusters	Maximum clusters to evaluate.

Value

Vector of CH index values.

calc_pac *Calculate Proportion of Ambiguous Clustering (PAC)*

Description

Computes PAC to evaluate clustering stability.

Usage

```
calc_pac(consensus_result, range_clusters = 2:6, x1 = 0.1, x2 = 0.9)
```

Arguments

consensus_result	Result from consensus_cluster().
range_clusters	Vector of K values to evaluate (default: 2:6).
x1	Lower bound threshold (default: 0.1).
x2	Upper bound threshold (default: 0.9).

Value

Data frame with PAC values for each K.

CalCHI *Calinski-Harabasz Index Calculation*

Description

Calculates the Calinski-Harabasz index for evaluating clustering quality.

Usage

```
CalCHI(
  hclust_result,
  dist_matrix = NULL,
  max_clusters = round(1 + 3.3 * log10(length(hclust_result$order)))
)
```

Arguments

hclust_result	A hierarchical clustering object (result of hclust).
dist_matrix	Optional distance matrix. If not provided, cophenetic is used.
max_clusters	Integer. Maximum number of clusters to evaluate.

Value

A vector containing CH index values for each number of clusters.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

CalPAC *Calculate Proportion of Ambiguous Clustering (PAC)*

Description

This function calculates the Proportion of Ambiguous Clustering (PAC) to help evaluate the optimal number of clusters in a consensus clustering analysis.

Usage

```
CalPAC(consensus_result, range_clusters = 2:6, x1 = 0.1, x2 = 0.9)
```

Arguments

consensus_result	A list containing consensus clustering results from ConsensusClusterPlus.
range_clusters	Integer vector. The range of cluster numbers evaluated during consensus clustering.
x1	Numeric. Lower bound for defining the PAC (default: 0.1).
x2	Numeric. Upper bound for defining the PAC (default: 0.9).

Details

The PAC is calculated as the difference between the cumulative distribution function values at two thresholds, x2 and x1, on the consensus matrix values. A lower PAC indicates a more stable clustering solution.

Value

A data frame containing the PAC values for each number of clusters.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
data <- mtcars
cc_res <- RunCC(data)
pac_values <- CalPAC(cc_res)
pac_values
```

---

check_sample_alignment

Check Sample Alignment

Description

Checks if samples are aligned across datasets.

Usage

```
check_sample_alignment(data_list, strict = FALSE)
```

Arguments

data_list	List of data matrices.
strict	If TRUE, requires identical sample names.

Value

TRUE if aligned, otherwise prints discrepancies.

cimlr_cluster *CIMLR Core Algorithm*

Description

Performs multi-kernel learning based clustering.

Usage

```
cimlr_cluster(data_list, k, n_kernels = 10, max_iter = 30)
```

Arguments

data_list	List of data matrices (features x samples).
k	Number of clusters.
n_kernels	Number of kernels per data type (default: 10).
max_iter	Maximum iterations (default: 30).

Value

List with clusters, kernel, weights.

cimlr_feature_ranking *CIMLR Feature Ranking*

Description

Ranks features based on their contribution to clustering.

Usage

```
cimlr_feature_ranking(data_list, cluster)
```

Arguments

data_list	List of data matrices.
cluster	Cluster assignments.

Value

List of feature importance for each data type.

`Classifier.Adaboost` *AdaBoost Classifier for Cluster Prediction*

Description

This function performs classification using AdaBoost to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.Adaboost(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains an AdaBoost model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.Adaboost(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.DT *Decision Tree Classifier for Cluster Prediction*

Description

This function performs classification using Decision Tree to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.DT(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a Decision Tree model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
```

```
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.DT(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.Enet` *Elastic Net Classifier for Cluster Prediction*

Description

This function performs classification using Elastic Net to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.Enet(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains an Elastic Net model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.Enet(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.Enrichment` *Enrichment-Based Neural Network Classifier for Cluster Prediction*

Description

This function performs classification using pathway enrichment analysis and a neural network to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.Enrichment(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE,
  nCores = 5
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names, 'OR' as odds ratio, and 'AUC' as area under curve.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.
`nCores`	An integer indicating the number of cores to use for pathway enrichment analysis. Default is 5.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Filters out markers with low odds ratio. 6. Performs pathway enrichment analysis using the ssMwwGST method.

Value

A data frame with: - ID: The sample identifier. - Cluster: The predicted cluster label for each sample. - NES: Normalized Enrichment Score (NES) for each cluster assignment.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.Enrichment(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.GBDT *Gradient Boosted Decision Trees (GBDT) Classifier for Cluster Prediction*

Description

This function performs classification using Gradient Boosted Decision Trees (GBDT) to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.GBDT(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a Gradient Boosted Decision Trees model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
```

```
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.GBDT(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.kNN` *k-Nearest Neighbors (kNN) Classifier for Cluster Prediction*

Description

This function performs classification using k-Nearest Neighbors to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.kNN(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.

`cluster.markers`
: A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.

`scale`
: A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names and matches training data. 2. Scales the test data for prediction if 'scale' is TRUE. 3. Selects genes that are common between the test and training datasets. 4. Uses glmnet to identify the important markers for each cluster and trains a k-Nearest Neighbors (kNN) model for classification. 5. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Cluster: The predicted cluster label for each sample. - Probabilities: The probabilities for each cluster assignment.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.kNN(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.LASSO` *LASSO Classifier for Cluster Prediction*

Description

This function performs classification using LASSO (Least Absolute Shrinkage and Selection Operator) to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.LASSO(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a LASSO model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.LASSO(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.LDA *Linear Discriminant Analysis (LDA) Classifier for Cluster Prediction*

Description

This function performs classification using Linear Discriminant Analysis (LDA) to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.LDA(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains an LDA model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
```

```
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.LDA(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.NBayes` *Naive Bayes Classifier for Cluster Prediction*

Description

This function performs classification using Naive Bayes to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.NBayes(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a Naive Bayes model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.NBayes(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.NNet` *Neural Network Classifier for Cluster Prediction*

Description

This function performs classification using a neural network to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.NNet(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a neural network model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.NNet(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.PCA *PCA-Based Neural Network Classifier for Cluster Prediction*

Description

This function performs classification using PCA and a neural network to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.PCA(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and performs PCA to reduce dimensionality. 6. Trains a neural network model for classification using the top PCs. 7. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
```

```
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.PCA(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.RF` *Random Forest Classifier for Cluster Prediction*

Description

This function performs classification using Random Forest to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.RF(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction if ‘scale‘ is TRUE. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a Random Forest model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Cluster: The predicted cluster label for each sample. - Probabilities: The probabilities for each cluster assignment.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.RF(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.Ridge` *Ridge Classifier for Cluster Prediction*

Description

This function performs classification using Ridge Regression to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.Ridge(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains a Ridge model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.Ridge(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.ssGSEA *Perform ssGSEA-based Subtyping Using Marker Gene Sets*

Description

This function performs single-sample Gene Set Enrichment Analysis (ssGSEA) to assign subtypes to samples based on marker gene sets. It calculates enrichment scores with permutation testing and predicts sample subtypes based on the most significant enrichment results.

Usage

```
Classifier.ssGSEA(
  data.test,
  marker.list,
  dir.file = ".",
  gct.filename = "data.gct",
```

```
  number.perms = 100,
  tolerate.mixed = FALSE,
  method = c("internal", "GSVA", "external"),
  seed = 12345
)
```

Arguments

data.test	A matrix or data frame representing the input expression data, where rows are genes and columns are samples.
marker.list	A named list of marker gene sets, where each list element corresponds to a specific subtype or category of interest.
dir.file	Character. Directory for saving the output files (default: '.'). Set to NULL to skip file output.
gct.filename	Character. The filename for the generated GCT file (default: 'data.gct').
number.perms	Integer. Number of permutations for ssGSEA analysis (default: 100).
tolerate.mixed	Logical. Whether to allow "Mixed" predictions when multiple gene sets have the same minimum p-value (default: FALSE).
method	Character. The ssGSEA implementation to use: "internal" (built-in), "GSVA" (requires GSVA package), or "external" (requires ssgsea.GBM.classification package). Default: "internal".
seed	Integer. Random seed for reproducibility (default: 12345).

Details

The function:

1. Calculates ssGSEA enrichment scores for each marker gene set in every sample.
2. Performs permutation testing to estimate statistical significance.
3. Predicts subtypes by identifying the marker gene set with the most significant enrichment (smallest p-value).
4. If tolerate.mixed is TRUE and multiple gene sets share the same minimum p-value, the sample is labeled as "Mixed".

Value

A data frame with the following columns:

- ID: Sample identifiers.
- Predict: Predicted subtype for each sample.
- Columns with _pval: P-values for each marker gene set or subtype.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Wang Q, Hu B, Hu X, Kim H, Squatrito M, Scarpace L, et al. Tumor Evolution of Glioma-Intrinsic Gene Expression Subtypes Associates with Immunological Changes in the Microenvironment. Cancer Cell. July 2017;32(1):42-56.e6.

Examples

```
# Simulated expression data
data.test <- matrix(rnorm(10000), nrow = 100, ncol = 100)
rownames(data.test) <- paste0("Gene", 1:100)
colnames(data.test) <- paste0("Sample", 1:100)

# Example marker list
marker.list <- list(
  Subtype1 = c("Gene1", "Gene2", "Gene3"),
  Subtype2 = c("Gene4", "Gene5", "Gene6")
)

# Run ssGSEA-based subtyping
result <- Classifier.ssGSEA(
  data.test = data.test,
  marker.list = marker.list,
  number.perms = 50,
  tolerate.mixed = TRUE
)
print(result)
```

Classifier.StepLR *Stepwise Logistic Regression Classifier for Cluster Prediction*

Description

This function performs classification using Stepwise Logistic Regression to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.StepLR(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Scales the test data for prediction. 3. Selects genes that are common between the test and training datasets. 4. Uses glmnet to identify the important markers for each cluster and trains a multinomial logistic regression model for classification. 5. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
```

```
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.StepLR(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.SVD` *SVD-Based Neural Network Classifier for Cluster Prediction*

Description

This function performs classification using SVD and a neural network to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.SVD(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and performs SVD to reduce dimensionality. 6. Trains a neural network model for classification using the top singular vectors. 7. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.SVD(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`Classifier.SVM` *Support Vector Machine (SVM) Classifier for Cluster Prediction*

Description

This function performs classification using Support Vector Machine (SVM) to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.SVM(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

`data.test`	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
`data.train`	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
`cluster.data`	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
`cluster.markers`	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
`scale`	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the ‘cluster.data‘ has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains an SVM model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.SVM(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

Classifier.XGBoost *XGBoost Classifier for Cluster Prediction*

Description

This function performs classification using XGBoost to predict cluster assignments for test data based on trained models from training data and cluster markers.

Usage

```
Classifier.XGBoost(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function operates as follows: 1. Ensures that the 'cluster.data' has the correct column names. 2. Adds a one-hot encoded matrix for cluster assignments. 3. Scales the test data for prediction. 4. Selects genes that are common between the test and training datasets. 5. Uses glmnet to identify the important markers for each cluster and trains an XGBoost model for classification. 6. Predicts the cluster for test samples and provides probabilities for each cluster.

Value

A data frame with: - ID: The sample identifier. - Probabilities: The probabilities for each cluster assignment. - Predict: The predicted cluster label for each sample.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
cluster.data <- data.frame(
  Sample = paste0("Sample", 1:60),
  Cluster = rep(paste0("C", 1:3), each = 20)
)
data.train <- matrix(rnorm(6000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), cluster.data$Sample)
)
data.test <- matrix(rnorm(5000),
  nrow = 100,
  dimnames = list(paste0("Gene", 1:100), paste0("P", 1:50))
)
cluster.markers <- setNames(
  lapply(
    unique(cluster.data$Cluster),
    function(cluster) {
      data.frame(Gene = sample(rownames(data.train), 10))
```

```
    }
  ),
  unique(cluster.data$Cluster)
)
result <- Classifier.XGBoost(
  data.test = data.test, data.train = data.train,
  cluster.data, cluster.markers
)
head(result)
```

`compare_clusterings` *Create Cluster Comparison Matrix*

Description

Creates a comparison matrix showing agreement between algorithms.

Usage

```
compare_clusterings(results)
```

Arguments

`results`	List of clustering results from run_multiple_algorithms().

Value

Matrix of Adjusted Rand Index values.

`compute_umap` *UMAP Dimensionality Reduction*

Description

Performs UMAP for visualization of multi-omics data.

Usage

```
compute_umap(data, n_neighbors = 15, min_dist = 0.1, n_epochs = 200, seed = 42)
```

Arguments

`data`	Data matrix (samples x features) or list of matrices.
`n_neighbors`	Number of neighbors (default: 15).
`min_dist`	Minimum distance (default: 0.1).
`n_epochs`	Number of epochs (default: 200).
`seed`	Random seed.

Value

Matrix with UMAP coordinates (samples x 2).

`consensus_cluster` *Consensus Clustering*

Description

Performs consensus clustering to identify stable clusters.

Usage

```
consensus_cluster(
  d,
  maxK = 6,
  reps = 1000,
  pItem = 0.8,
  pFeature = 1,
  clusterAlg = "hc",
  innerLinkage = "ward.D2",
  finalLinkage = "ward.D2",
  distance = "euclidean",
  seed = NULL,
  verbose = FALSE
)
```

Arguments

`d`	Data matrix (features x samples) or distance object.
`maxK`	Maximum number of clusters to evaluate (default: 6).
`reps`	Number of resampling iterations (default: 1000).
`pItem`	Proportion of items to sample in each iteration (default: 0.8).
`pFeature`	Proportion of features to sample (default: 1).
`clusterAlg`	Clustering algorithm: "hc", "km", or "pam" (default: "hc").
`innerLinkage`	Linkage method for hierarchical clustering (default: "ward.D2").
`finalLinkage`	Linkage for final clustering (default: "ward.D2").
`distance`	Distance metric (default: "euclidean").
`seed`	Random seed for reproducibility.
`verbose`	Print progress messages.

Value

List containing consensus matrices and clustering results.

correct_batch *Simple Batch Correction*

Description

Performs simple batch correction using mean centering.

Usage

```
correct_batch(data_list, batch, method = "center")
```

Arguments

data_list	List of data matrices.
batch	Vector of batch labels for each sample.
method	Method: "center" (mean centering) or "combat_simple".

Value

List of batch-corrected matrices.

cpca *Consensus PCA*

Description

Performs Consensus PCA on multiple data types.

Usage

```
cpca(data_list, ncomp = 2)
```

Arguments

data_list	List of data matrices (features x samples).
ncomp	Number of components (default: 2).

Value

List with scores, loadings, eigenvalues.

CV *Calculate Coefficient of Variation for a Numeric Vector*

Description

This function calculates the coefficient of variation (CV) for a given numeric vector.

Usage

```
CV(V)
```

Arguments

V	A numeric vector.

Value

The coefficient of variation (CV) for the input vector.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

CV.df *Calculate Coefficient of Variation for a Data Frame*

Description

This function calculates the coefficient of variation (CV) for each column in a data frame.

Usage

```
CV.df(df)
```

Arguments

df	A data frame with row samples and column features.

Value

A sorted vector of CV values for each column in the data frame, in decreasing order.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

`FeatureSelectionWithBootstrap` *Feature Selection with Bootstrap for Each Cluster*

Description

This function performs feature selection using a logistic regression model and bootstrapping for each cluster in the dataset. For each cluster, the function evaluates which features are significantly associated with the cluster based on a logistic regression model with bootstrap sampling. The output includes a count of significant features for each cluster based on the p-value threshold.

Usage

```
FeatureSelectionWithBootstrap(
  data,
  p.no_bootstrap = 0.01,
  p.bootstrap = 0.05,
  num.iteration = 1000,
  nCores = parallel::detectCores() - 3
)
```

Arguments

`data`	A data frame where the first column is the cluster variable (categorical), and the other columns are feature values.
`p.no_bootstrap`	Numeric. The p-value threshold for the features to be selected based on the original data (default: 0.01).
`p.bootstrap`	Numeric. The p-value threshold for the features to be selected based on bootstrapped samples (default: 0.05).
`num.iteration`	Integer. The number of bootstrap iterations (default: 1000).
`nCores`	Integer. The number of CPU cores to use for parallel processing (default: automatically set to all cores minus 3).

Value

A list of data frames, each containing the features selected for each cluster and the count of significant features based on bootstrapping.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

filter_by_mad *Filter by MAD (Median Absolute Deviation)*

Description

Keeps features with highest MAD values.

Usage

```
filter_by_mad(data_list, top_n = 5000)
```

Arguments

data_list	List of data matrices.
top_n	Number of top features to keep.

Value

List of filtered matrices.

filter_low_variance *Filter Low-Variance Features*

Description

Removes features with low variance across samples.

Usage

```
filter_low_variance(data_list, min_var = 0.01, top_n = NULL, top_pct = NULL)
```

Arguments

data_list	List of data matrices (features x samples).
min_var	Minimum variance threshold (default: 0.01).
top_n	Keep top N features by variance (optional).
top_pct	Keep top percentage of features (optional, 0-1).

Value

List of filtered matrices.

`Find.OptClusterFeatures` *Optimal Feature Combination for Multi-Modality Clustering*

Description

Selects optimal feature combinations for multi-modality clustering analysis.

Usage

```
Find.OptClusterFeatures(
  data_layers,
  feature_subset_sizes,
  try_num_clusters = 2:6,
  n_runs = 5,
  n_fold = 5
)
```

Arguments

`data_layers`	A named list of matrices (features x samples).
`feature_subset_sizes`	A list of sequences for feature subset sizes.
`try_num_clusters`	Integer vector. Cluster numbers to test (default: 2:6).
`n_runs`	Integer. NMF iterations (default: 5).
`n_fold`	Integer. Cross-validation folds (default: 5).

Value

A list with optimal_combination and all_results.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

gene_sets *Functional Gene Sets*

Description

The gene_sets data object contains a unified collection of gene sets derived from multiple sources, including curated pathways (c2.cp.v2022.1.Hs), Gene Ontology terms (c5.go.v2022.1.Hs), and hallmark gene sets (h.all.v2022.1.Hs). These gene sets are based on version 2022.1 and represent key biological processes, pathways, and well-defined biological states. By integrating these sources, the gene_sets object provides a comprehensive dataset that can be utilized for enrichment analysis and functional exploration, offering valuable insights into underlying biological mechanisms.

Usage

```
gene_sets
```

Format

data.frame

get.binary.clusters *Get Binary Clusters from Clustering Results*

Description

This function extracts binary cluster assignments from multiple clustering results.

Usage

```
get.binary.clusters(res)
```

Arguments

res	A list containing clustering results

Value

A data frame where each row represents a binary encoding of cluster assignments across different methods.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

get.class *Get Cluster Assignments*

Description

Extract cluster assignments from Consensus Clustering results for a specific number of clusters.

Usage

```
get.class(cc.res, k)
```

Arguments

cc.res	Consensus clustering results from ConsensusClusterPlus.
k	Integer. The number of clusters to extract.

Value

A data frame with the following columns: - ID: The sample identifier. - Cluster: The assigned cluster label, prefixed by 'C'.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
data <- mtcars
cc_res <- RunCC(data)
clu <- get.class(cc_res, 2)
clu
```

get.Jaccard.Distance *Jaccard Distance Calculation for Binary Matrix*

Description

This function calculates the Jaccard distance or similarity for a binary matrix. It is typically used to evaluate the similarity or dissimilarity between columns of a binary matrix.

Usage

```
get.Jaccard.Distance(data, dissimilarity = TRUE)
```

Arguments

data	A binary matrix where rows represent features and columns represent samples.
dissimilarity	Logical. If TRUE, returns the Jaccard distance; if FALSE, returns the Jaccard similarity (default: TRUE).

Details

The function computes the Jaccard distance (or similarity) between each pair of columns in the input binary matrix. The Jaccard distance is calculated as 1 minus the Jaccard similarity.

Value

A matrix containing the Jaccard distance or similarity between each pair of columns in the input matrix.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
data <- matrix(sample(0:1, 1500, replace = TRUE), nrow = 30, ncol = 50)
jaccard_dist <- get.Jaccard.Distance(as.data.frame(data), dissimilarity = TRUE)
jaccard_dist
```

---

get_consensus_class *Get Cluster Assignments from Consensus Results*

Description

Extracts cluster assignments for a specific K.

Usage

```
get_consensus_class(consensus_fit, k)
```

Arguments

consensus_fit	Result from consensus_cluster().
k	Number of clusters.

Value

Data frame with sample IDs and cluster assignments.

handle_missing *Handle Missing Values*

Description

Imputes or removes missing values in multi-omics data.

Usage

```
handle_missing(data_list, method = "median", threshold = 0.5, k = 5)
```

Arguments

data_list	List of data matrices.
method	Method: "remove_features", "remove_samples", "mean", "median", "knn".
threshold	For remove methods: maximum proportion of NA allowed.
k	For KNN imputation: number of neighbors.

Value

List of processed matrices.

icluster_bayes_fast *Simplified iClusterBayes*

Description

Faster version using variational approximation.

Usage

```
icluster_bayes_fast(data_list, k, max_iter = 100, tol = 1e-04, verbose = FALSE)
```

Arguments

data_list	List of data matrices.
k	Number of clusters.
max_iter	Maximum iterations (default: 100).
tol	Convergence tolerance (default: 1e-4).
verbose	Print progress.

Value

List with cluster assignments and parameters.

init *Initialize MOFSR Optional Dependencies*

Description

Installs optional packages for extended functionality.

Usage

```
init(classifier = TRUE, survival = FALSE, gsva = FALSE)
```

Arguments

classifier	Logical. Install classifier dependencies (default: TRUE).
survival	Logical. Install survival analysis dependencies (default: FALSE).
gsva	Logical. Install GSVA dependencies (default: FALSE).

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

intnmf_cluster *IntNMF Core Algorithm*

Description

Main IntNMF algorithm using non-negative alternating least squares.

Usage

```
intnmf_cluster(
  dat,
  k,
  maxiter = 200,
  st_count = 20,
  n_ini = 30,
  ini_nndsvd = TRUE,
  seed = TRUE,
  wt = NULL
)
```

Arguments

dat	List of data matrices (samples x features).
k	Number of clusters.
maxiter	Maximum iterations (default: 200).
st_count	Stability count for convergence (default: 20).
n_ini	Number of initializations (default: 30).
ini_nndsvd	Use NNDSVD initialization (default: TRUE).
seed	Use random seed (default: TRUE).
wt	Weights for each data set.

Value

List with W, H matrices, clusters, and convergence info.

---

intnmf_opt_k *Optimal K Selection for IntNMF*

Description

Selects optimal number of clusters using cross-validation.

Usage

```
intnmf_opt_k(
  dat,
  n_runs = 30,
  n_fold = 5,
  k_range = 2:8,
  maxiter = 100,
  st_count = 10,
  wt = NULL,
  verbose = TRUE
)
```

Arguments

dat	List of data matrices.
n_runs	Number of runs (default: 30).
n_fold	Number of CV folds (default: 5).
k_range	Range of K values to test (default: 2:8).
maxiter	Maximum iterations (default: 100).
st_count	Stability count (default: 10).
wt	Weights for datasets.
verbose	Print progress.

Value

Matrix of CPI values for each K and run.

`lracluster` *LRAcluster Core Algorithm*

Description

Performs low-rank approximation clustering.

Usage

```
lracluster(data, types, dimension = 2, names = NULL)
```

Arguments

`data`	List of data matrices.
`types`	Character vector of data types ("binary", "gaussian", "poisson").
`dimension`	Target dimension/rank (default: 2).
`names`	Names for each dataset.

Value

List with coordinate matrix and potential (quality measure).

`MAD.df` *Calculate Median Absolute Deviation for a Data Frame*

Description

This function calculates the median absolute deviation (MAD) for each column in a data frame.

Usage

```
MAD.df(df)
```

Arguments

`df`	A data frame with row samples and column features.

Value

A sorted vector of MAD values for each column in the data frame, in decreasing order.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

mcia *Multiple Co-Inertia Analysis*

Description

Performs MCIA on multiple data matrices.

Usage

```
mcia(data_list, ncomp = 2)
```

Arguments

data_list	List of data matrices (features x samples).
ncomp	Number of components (default: 2).

Value

List with global scores, block scores, loadings, and eigenvalues.

Mean.df *Calculate Mean Value for a Data Frame*

Description

This function calculates the mean value for each column in a data frame.

Usage

```
Mean.df(df)
```

Arguments

df	A data frame with row samples and column features.

Value

A vector of mean values for each column in the data frame.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

Median.df *Calculate Median Value for a Data Frame*

Description

This function calculates the median value for each column in a data frame.

Usage

```
Median.df(df)
```

Arguments

df	A data frame with row samples and column features.

Value

A vector of median values for each column in the data frame.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

minmax *Minmax Normalization for a Numeric Vector*

Description

This function performs min-max normalization on a numeric vector.

Usage

```
minmax(x)
```

Arguments

x	A numeric vector.

Value

A normalized numeric vector with values between 0 and 1.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

minmax.df *Minmax Normalization for a Data Frame*

Description

This function performs min-max normalization on each column in a data frame.

Usage

```
minmax.df(data)
```

Arguments

data	A data frame with row samples and column features.

Value

A data frame with normalized values between 0 and 1 for each column.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

nemo_affinity_graph *NEMO Affinity Graph Construction*

Description

Constructs a single affinity graph from multi-omics data.

Usage

```
nemo_affinity_graph(raw_data, k = NA)
```

Arguments

raw_data	List of data matrices (features x samples).
k	Number of neighbors. Can be a number, vector, or NA.

Value

Affinity matrix measuring similarity across all omics.

nemo_clustering *NEMO Clustering*

Description

Performs multi-omic clustering using the NEMO algorithm.

Usage

```
nemo_clustering(omics_list, num_clusters = NULL, num_neighbors = NA)
```

Arguments

omics_list	List of data matrices (features x samples).
num_clusters	Number of clusters (NULL for automatic estimation).
num_neighbors	Number of neighbors (NA for automatic selection).

Value

Named vector of cluster assignments.

nemo_num_clusters *Estimate Number of Clusters from Affinity Graph*

Description

Uses eigengap heuristic to estimate optimal cluster number.

Usage

```
nemo_num_clusters(W, NUMC = 2:15)
```

Arguments

W	Affinity matrix.
NUMC	Possible cluster numbers to evaluate (default: 2:15).

Value

Estimated number of clusters.

parallel *Parallel Computing Support for MOFSR*

Description

Functions for parallel execution of multi-omics analysis.

Sets up parallel processing using the future framework.

Usage

```
setup_parallel(workers = NULL, strategy = "multisession", verbose = TRUE)
```

Arguments

workers	Number of worker processes (default: detectCores() - 1).
strategy	Parallel strategy: "multisession", "multicore", or "sequential".
verbose	Print configuration info (default: TRUE).

Details

- "multisession": Works on all platforms (Windows, macOS, Linux) - "multicore": Faster on Unix-like systems, not available on Windows - "sequential": No parallelism (useful for debugging)

Value

Invisibly returns the previous plan.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

parallel_bootstrap_features *Parallel Bootstrap Feature Selection*

Description

Performs bootstrap-based feature selection in parallel.

Usage

```
parallel_bootstrap_features(
  data,
  n_bootstrap = 1000,
  n_workers = NULL,
  p_threshold = 0.05,
  progress = TRUE
)
```

Arguments

data	Data frame with cluster variable in first column.
n_bootstrap	Number of bootstrap iterations (default: 1000).
n_workers	Number of workers (default: auto).
p_threshold	P-value threshold (default: 0.05).
progress	Show progress (default: TRUE).

Value

List of significant features per cluster.

---

parallel_consensus_cluster

Parallel Consensus Clustering

Description

Performs consensus clustering with parallel resampling.

Usage

```
parallel_consensus_cluster(
  data,
  max_k = 6,
  n_reps = 1000,
  n_workers = NULL,
  prop_sample = 0.8,
  cluster_method = "hclust",
  verbose = TRUE
)
```

Arguments

data	Data matrix (features x samples).
max_k	Maximum number of clusters (default: 6).
n_reps	Number of resampling iterations (default: 1000).
n_workers	Number of workers (default: auto).
prop_sample	Proportion of samples to include (default: 0.8).
cluster_method	Base clustering method (default: "hclust").
verbose	Print progress (default: TRUE).

Value

List with consensus matrices and cluster results.

PathDEA *Pathway Differential Expression Analysis (PathDEA)*

Description

This function performs pathway differential expression analysis based on clustering results and pathway activity scores derived from ssMwwGST.

Usage

```
PathDEA(
  Cluster_data,
  ssMwwGST_results,
  dea_FDR_threshold = 0.001,
  dea_gap_threshold = 1.5
)
```

Arguments

Cluster_data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments.
ssMwwGST_results	A list of results from ssMwwGST, including NES (Normalized Enrichment Scores).
dea_FDR_threshold	Numeric. The FDR threshold to use for filtering significant pathways. Default is 0.001.
dea_gap_threshold	Numeric. The median gap threshold to use for filtering significant pathways. Default is 1.5.

Details

The function operates as follows: 1. Extracts Normalized Enrichment Scores (NES) from the ssMwwGST results. 2. Performs Wilcoxon rank-sum tests to compare pathway activity between clusters for each pathway. 3. Calculates median and mean differences in pathway activity between clusters. 4. Adjusts p-values using the Benjamini-Hochberg method to control the false discovery rate (FDR).

Value

A list containing: - dea_path: A list of data frames, each containing the differential expression analysis results for each cluster. - dea_path2: A list of data frames containing the filtered differential expression analysis results for each cluster based on FDR and median gap thresholds. - NES: A data frame of Normalized Enrichment Scores for each gene set and each sample. - Cluster: A data frame of sample IDs and their corresponding cluster assignments.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
# Example usage:
Cluster_data <- data.frame(Sample = paste0("Sample", 1:10), Cluster = rep(1:2, each = 5))
ssMwwGST_results <- list(NES = matrix(rnorm(100),
  nrow = 10, ncol = 10,
  dimnames = list(paste0("Pathway", 1:10), paste0("Sample", 1:10))
))
result <- PathDEA(Cluster_data, ssMwwGST_results)
```

---

perturbation_clustering

Perturbation Clustering

Description

Performs perturbation-based clustering to find optimal k.

Usage

```
perturbation_clustering(
  data,
  kMin = 2,
  kMax = 5,
  k = NULL,
  n_iter = 50,
  clustering_method = "kmeans",
  verbose = FALSE
)
```

Arguments

data	Data matrix (samples x features).
kMin	Minimum number of clusters (default: 2).
kMax	Maximum number of clusters (default: 5).
k	Fixed number of clusters (optional).
n_iter	Number of perturbation iterations (default: 50).
clustering_method	Clustering method: "kmeans", "hclust", "pam".
verbose	Print progress.

Value

List with k, cluster, origS, pertS.

---

plot_algorithm_comparison

Plot Algorithm Comparison

Description

Visualizes agreement between different clustering algorithms.

Usage

```
plot_algorithm_comparison(results, title = "Algorithm Agreement (ARI)")
```

Arguments

results	List of clustering results from run_multiple_algorithms().
title	Plot title.

Value

A ggplot object or base R plot.

plot_cluster_quality *Plot Cluster Quality Metrics*

Description

Visualizes PAC, CHI, and silhouette scores across different K values.

Usage

```
plot_cluster_quality(
  pac_values,
  chi_values = NULL,
  title = "Cluster Quality Assessment"
)
```

Arguments

pac_values	PAC values from CalPAC().
chi_values	CHI values from CalCHI() (optional).
title	Plot title.

Value

A ggplot object or base R plot.

plot_consensus_heatmap *Plot Consensus Matrix Heatmap*

Description

Visualizes the consensus matrix from consensus clustering.

Usage

```
plot_consensus_heatmap(
  consensus_matrix,
  clusters = NULL,
  title = "Consensus Matrix",
  colors = NULL
)
```

Arguments

consensus_matrix	Consensus matrix (from RunCC or consensus_cluster).
clusters	Vector of cluster assignments for annotation.
title	Plot title.
colors	Color palette for heatmap.

Value

Invisibly returns the plot object.

---

plot_silhouette *Plot Silhouette Analysis*

Description

Creates a silhouette plot for cluster quality evaluation.

Usage

```
plot_silhouette(
  clusters,
  dist_matrix,
  title = "Silhouette Analysis",
  colors = NULL
)
```

Arguments

clusters	Vector of cluster assignments.
dist_matrix	Distance matrix.
title	Plot title.
colors	Cluster colors.

Value

A ggplot object or base R plot.

plot_survival *Plot Kaplan-Meier Survival Curves*

Description

Visualizes survival differences between clusters.

Usage

```
plot_survival(
  time,
  event,
  clusters,
  title = "Kaplan-Meier Survival Curves",
  colors = NULL,
  conf_int = TRUE,
  risk_table = FALSE
)
```

Arguments

time	Survival time vector.
event	Event indicator (1 = event, 0 = censored).
clusters	Vector of cluster assignments.
title	Plot title.
colors	Cluster colors.
conf_int	Show confidence intervals (default: TRUE).
risk_table	Show risk table (default: FALSE).

Value

A ggplot object (if survminer available) or base R plot.

plot_umap *Plot UMAP with Clusters*

Description

Creates a UMAP plot colored by cluster assignments.

Usage

```
plot_umap(
  umap_coords,
  clusters,
  title = "UMAP Visualization",
  point_size = 2,
  colors = NULL,
  show_legend = TRUE
)
```

Arguments

umap_coords	UMAP coordinates from compute_umap().
clusters	Vector of cluster assignments or data frame with Cluster column.
title	Plot title (default: "UMAP Visualization").
point_size	Point size (default: 2).
colors	Custom color palette (optional).
show_legend	Show legend (default: TRUE).

Value

A ggplot object if ggplot2 is available, otherwise base R plot.

---

preprocessing *Data Preprocessing Functions for MOFSR*

Description

Functions for preprocessing multi-omics data before integration.

Applies normalization to each data matrix in the list.

Usage

```
normalize_omics(data_list, method = "zscore", by_feature = TRUE)
```

Arguments

data_list	List of data matrices (features x samples).
method	Normalization method: "zscore", "minmax", "quantile", "log2", "vst".
by_feature	Normalize by feature (row) or sample (column).

Value

List of normalized matrices.

Author(s)

Zaoqu Liu

`qc_summary` *Quality Control Summary*

Description

Generates quality control summary for multi-omics data.

Usage

```
qc_summary(data_list)
```

Arguments

`data_list`	List of data matrices.

Value

Data frame with QC metrics for each dataset.

`rgcca` *Regularized Generalized Canonical Correlation Analysis*

Description

Performs RGCCA for multi-block data analysis.

Usage

```
rgcca(
  A,
  C = 1 - diag(length(A)),
  tau = rep(1, length(A)),
  ncomp = rep(1, length(A)),
  scheme = "centroid",
  scale = TRUE,
  init = "svd",
  bias = TRUE,
  tol = 1e-08,
  verbose = FALSE
)
```

Arguments

A	List of data blocks (samples x variables).
C	Design matrix (default: complete design).
tau	Shrinkage parameters (default: 1 for each block).
ncomp	Number of components per block (default: 1).
scheme	Scheme: "horst", "factorial", or "centroid".
scale	Scale blocks (default: TRUE).
init	Initialization: "svd" or "random".
bias	Biased estimator (default: TRUE).
tol	Convergence tolerance.
verbose	Print progress.

Value

List with Y, a, astar, C, tau, scheme, ncomp, crit, AVE.

---

run-clustering *Unified Multi-Omics Clustering Interface*

Description

Provides a unified interface for all clustering algorithms in MOFSR.

Returns available multi-omics clustering algorithms.

Usage

```
list_clustering_algorithms()
```

Value

Character vector of algorithm names.

Author(s)

Zaoqu Liu

run_bcc *Run BCC Clustering*

Description

Performs BCC-based clustering on multi-omics data.

Usage

```
run_bcc(data, n_clusters, method = "fast", n_iter = 1000)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
method	Method: "full" for full Bayesian, "fast" for EM-like.
n_iter	MCMC iterations (for full method).

Value

Data frame with sample IDs and cluster assignments.

run_cimlr *Run CIMLR Clustering*

Description

Performs CIMLR-based clustering on multi-omics data.

Usage

```
run_cimlr(data, n_clusters, n_kernels = 10)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
n_kernels	Number of kernels per data type (default: 10).

Value

Data frame with sample IDs and cluster assignments.

run_cpca *Run CPCA Clustering*

Description

Performs CPCA-based clustering on multi-omics data.

Usage

```
run_cpca(data, n_clusters, ncomp = NULL, cluster_method = "ward.D2")
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
ncomp	Number of components.
cluster_method	Clustering method.

Value

Data frame with cluster assignments.

run_iclusterbayes *Run iClusterBayes Clustering*

Description

Performs iClusterBayes-based clustering on multi-omics data.

Usage

```
run_iclusterbayes(
  data,
  n_clusters,
  method = "fast",
  n_burnin = 500,
  n_sample = 1000
)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
method	Method: "full" for full Bayesian, "fast" for variational.
n_burnin	Burn-in iterations (for full method).
n_sample	Sampling iterations (for full method).

Value

Data frame with sample IDs and cluster assignments.

`run_integration` *Run Multi-Omics Integration and Clustering*

Description

Unified function to run any multi-omics clustering algorithm.

Usage

```
run_integration(data, algorithm, n_clusters, ...)
```

Arguments

`data`	List of matrices (features x samples). All matrices must have the same samples (columns).
`algorithm`	Clustering algorithm name. Use list_clustering_algorithms() to see available options.
`n_clusters`	Number of clusters.
`...`	Additional arguments passed to specific algorithm.

Value

Data frame with sample IDs, cluster assignments, and cluster labels.

Examples

```
## Not run:
# Create example data
data1 <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
data2 <- matrix(rnorm(80 * 50), nrow = 80, ncol = 50)
colnames(data1) <- colnames(data2) <- paste0("Sample", 1:50)

data_list <- list(GE = data1, ME = data2)

# Run SNF clustering
result <- run_integration(data_list, "SNF", n_clusters = 3)

## End(Not run)
```

run_intnmf *Run IntNMF Clustering*

Description

High-level function for IntNMF clustering on multi-omics data.

Usage

```
run_intnmf(data, n_clusters, maxiter = 200, n_init = 30, weight = NULL)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
maxiter	Maximum iterations (default: 200).
n_init	Number of initializations (default: 30).
weight	Weights for each data type (default: equal weights).

Value

Data frame with sample IDs and cluster assignments.

run_late_fusion *Late Fusion Clustering*

Description

Ensemble clustering by combining single-omics results.

Usage

```
run_late_fusion(
  data,
  n_clusters,
  single_method = "kmeans",
  consensus_method = "similarity"
)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
single_method	Method for single-omics clustering ("kmeans", "hclust", "pam").
consensus_method	Method for combining results ("voting", "similarity").

Value

Data frame with cluster assignments.

`run_lracluster` *Run LRAcluster*

Description

Performs LRAcluster-based clustering on multi-omics data.

Usage

```
run_lracluster(data, n_clusters, types = NULL, cluster_method = "ward.D2")
```

Arguments

`data`	List of matrices (features x samples).
`n_clusters`	Number of clusters.
`types`	Character vector of data types.
`cluster_method`	Hierarchical clustering method (default: "ward.D2").

Value

Data frame with sample IDs and cluster assignments.

`run_mcia` *Run MCIA Clustering*

Description

Performs MCIA-based clustering on multi-omics data.

Usage

```
run_mcia(data, n_clusters, ncomp = NULL, cluster_method = "ward.D2")
```

Arguments

`data`	List of matrices (features x samples).
`n_clusters`	Number of clusters.
`ncomp`	Number of MCIA components (default: auto).
`cluster_method`	Clustering method (default: "ward.D2").

Value

Data frame with sample IDs and cluster assignments.

run_mofa *Run Factor-Based Clustering*

Description

Performs factor analysis based clustering on multi-omics data. Uses multi-view factor analysis to extract latent factors, then clusters samples based on these factors. For full MOFA, use MOFA2 package.

Usage

```
run_mofa(data, n_clusters, n_factors = 10, cluster_method = "ward.D2")
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
n_factors	Number of factors (default: 10).
cluster_method	Clustering method (default: "ward.D2").

Value

Data frame with sample IDs and cluster assignments.

run_multiple_algorithms *Run Multiple Clustering Algorithms*

Description

Runs multiple algorithms and returns combined results.

Usage

```
run_multiple_algorithms(data, algorithms = NULL, n_clusters, ...)
```

Arguments

data	List of data matrices.
algorithms	Character vector of algorithm names.
n_clusters	Number of clusters.
...	Additional arguments.

Value

List of results for each algorithm.

run_nemo *Run NEMO Clustering*

Description

Performs NEMO-based clustering on multi-omics data.

Usage

```
run_nemo(data, n_clusters = NULL, n_neighbors = NA)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters (NULL for automatic).
n_neighbors	Number of neighbors (NA for automatic).

Value

Data frame with sample IDs and cluster assignments.

---

run_parallel_algorithms *Run Multiple Algorithms in Parallel*

Description

Executes multiple clustering algorithms in parallel.

Usage

```
run_parallel_algorithms(
  data,
  algorithms = NULL,
  n_clusters,
  n_workers = NULL,
  ...
)
```

Arguments

data	List of data matrices (features x samples).
algorithms	Character vector of algorithm names.
n_clusters	Number of clusters.
n_workers	Number of parallel workers (default: auto).
...	Additional arguments passed to algorithms.

Value

List of results for each algorithm.

`run_pinsplus` *Run PINSPlus Clustering*

Description

Performs PINSPlus-based clustering on multi-omics data.

Usage

```
run_pinsplus(data, n_clusters = NULL, kMin = 2, kMax = 5)
```

Arguments

`data`	List of matrices (features x samples).
`n_clusters`	Number of clusters (optional).
`kMin`	Minimum clusters (default: 2).
`kMax`	Maximum clusters (default: 5).

Value

Data frame with sample IDs and cluster assignments.

`run_rgcca` *Run RGCCA Clustering*

Description

Performs RGCCA-based clustering on multi-omics data.

Usage

```
run_rgcca(
  data,
  n_clusters,
  ncomp = NULL,
  tau = NULL,
  scheme = "centroid",
  cluster_method = "ward.D2"
)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
ncomp	Number of components (default: 1 per block).
tau	Shrinkage parameters.
scheme	Scheme type (default: "centroid").
cluster_method	Hierarchical clustering method (default: "ward.D2").

Value

Data frame with sample IDs and cluster assignments.

---

run_sgcca *Run SGCCA Clustering*

Description

Performs SGCCA-based clustering on multi-omics data.

Usage

```
run_sgcca(
  data,
  n_clusters,
  ncomp = NULL,
  c1 = NULL,
  scheme = "centroid",
  cluster_method = "ward.D2"
)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
ncomp	Number of components.
c1	L1 penalty parameters.
scheme	Scheme type.
cluster_method	Clustering method.

Value

Data frame with cluster assignments.

run_snf *Run SNF Clustering*

Description

Performs SNF-based clustering on multi-omics data.

Usage

```
run_snf(data, n_clusters, n_neighbors = 20, sigma = 0.5, n_iterations = 20)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
n_neighbors	Number of neighbors for affinity matrix (default: 20).
sigma	Variance for local model (default: 0.5).
n_iterations	Number of SNF iterations (default: 20).

Value

Data frame with sample IDs, cluster assignments, and cluster labels.

run_wsnf *Weighted Similarity Network Fusion*

Description

SNF with custom weights for each data type.

Usage

```
run_wsnf(
  data,
  n_clusters,
  weights = NULL,
  n_neighbors = 20,
  sigma = 0.5,
  n_iterations = 20
)
```

Arguments

data	List of matrices (features x samples).
n_clusters	Number of clusters.
weights	Numeric vector of weights for each data type.
n_neighbors	Number of neighbors (default: 20).
sigma	Kernel bandwidth (default: 0.5).
n_iterations	SNF iterations (default: 20).

Value

Data frame with cluster assignments.

RunBCC *Run Bayesian Consensus Clustering (BCC)*

Description

Performs multi-omics clustering using BCC algorithm.

Usage

```
RunBCC(data, N.clust, method = "fast", max.iterations = 1000)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
method	Character. "fast" or "full" (default: "fast").
max.iterations	Integer. Maximum iterations (default: 1000).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Lock EF, Dunson DB. Bioinformatics. 2013.

RunCC *Run Consensus Clustering*

Description

Performs consensus clustering on the given data.

Usage

```
RunCC(
  data,
  maxK = 6,
  reps = 1000,
  pItem = 0.8,
  pFeature = 1,
  clusterAlg = "hc",
  distance = "euclidean",
  innerLinkage = "ward.D2",
  finalLinkage = "ward.D2",
  seed = 1234,
  verbose = FALSE
)
```

Arguments

data	A numeric matrix (features x samples).
maxK	Maximum number of clusters (default: 6).
reps	Number of subsamples (default: 1000).
pItem	Proportion of items to sample (default: 0.8).
pFeature	Proportion of features to sample (default: 1).
clusterAlg	Clustering algorithm: "hc", "km", or "pam" (default: "hc").
distance	Distance metric (default: "euclidean").
innerLinkage	Linkage method for HC (default: "ward.D2").
finalLinkage	Linkage for final clustering (default: "ward.D2").
seed	Random seed (default: 1234).
verbose	Print progress (default: FALSE).

Value

A list containing consensus clustering results.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

RunCIMLR *Run Cancer Integrative Multi-kernel Learning (CIMLR)*

Description

Performs multi-omics clustering using CIMLR algorithm.

Usage

```
RunCIMLR(data, N.clust, n.kernels = 10)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
n.kernels	Integer. Number of kernels per data type (default: 10).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Ramazzotti D, et al. Nature Communications. 2018.

RunClassifier *Run Classifiers for Cluster Prediction*

Description

This function runs different classification models based on user input to predict cluster assignments for test data.

Usage

```
RunClassifier(
  algorithm,
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  scale = TRUE
)
```

Arguments

algorithm	A character string indicating the classifier to use. Supported algorithms include: "Adaboost", "DT", "Enet", "Enrichment", "GBDT", "LASSO", "LDA", "NBayes", "NNet", "PCA", "Ridge", "StepLR", "SVD", "SVM", "XGBoost", "kNN", "RF".
data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
scale	A logical value indicating whether to scale the test data. Default is TRUE.

Details

The function dynamically selects and runs a classification model based on user input. The supported classifiers include a range of machine learning models such as Random Forest, kNN, PCA, SVM, LASSO, Ridge, and others.

Value

A data frame containing the prediction results based on the selected algorithm.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
# Example usage:
data.test <- matrix(rnorm(1000), nrow = 100, ncol = 10)
data.train <- matrix(rnorm(1000), nrow = 100, ncol = 10)
cluster.data <- data.frame(Sample = paste0("Sample", 1:10), Cluster = rep(1:2, each = 5))
cluster.markers <- setNames(lapply(unique(cluster.data$Cluster), function(c) data.frame(Gene = paste0("Gene", sam
result <- RunClassifier(algorithm = "RF", data.test, data.train, cluster.data, cluster.markers, scale = TRUE)
```

RunCOCA *Run Consensus Clustering Analysis (COCA)*

Description

Performs Consensus Clustering Analysis to identify stable clusters.

Usage

```
RunCOCA(
  jaccard.matrix,
  max.clusters = 6,
  optimal.clusters = 3,
  linkage.method = "ward.D2",
  clustering.algorithm = "hc",
  distance.metric = "euclidean",
  resampling.iterations = 10000,
  resample.proportion = 0.7
)
```

Arguments

jaccard.matrix	A Jaccard distance matrix.
max.clusters	Integer. Maximum number of clusters (default: 6).
optimal.clusters	Integer. Optimal number of clusters (default: 3).
linkage.method	Character. Linkage method (default: "ward.D2").
clustering.algorithm	Character. Clustering algorithm (default: "hc").
distance.metric	Character. Distance metric (default: "euclidean").
resampling.iterations	Integer. Resampling iterations (default: 10000).
resample.proportion	Numeric. Resample proportion (default: 0.7).

Value

A list with consensus clustering results.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

RunCPCA *Run Consensus Principal Component Analysis (CPCA)*

Description

Performs multi-omics clustering using CPCA algorithm.

Usage

```
RunCPCA(data, N.clust, ncomp = NULL, clustering.algorithm = "ward.D2")
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
ncomp	Integer. Number of components (default: auto).
clustering.algorithm	Character. Clustering method (default: "ward.D2").

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

RunEnsemble *Run Ensemble of Multiple Classifiers for Cluster Prediction*

Description

This function runs an ensemble of different classification models to predict cluster assignments for test data, considering consensus among the models.

Usage

```
RunEnsemble(
  data.test,
  data.train,
  cluster.data,
  cluster.markers,
  surdata = NULL,
  time = "time",
  event = "event",
```

```
  methods = NULL,
  sur.trend.rank = NULL,
  cutoff.P = 0.05
)
```

Arguments

data.test	A numeric matrix or data frame of test data. Rows represent genes, and columns represent samples.
data.train	A numeric matrix or data frame of training data. Rows represent genes, and columns represent samples.
cluster.data	A data frame where the first column must be the sample IDs and the second column must be the cluster assignments. The sample IDs must match the column names of the training data.
cluster.markers	A list of data frames, each containing markers for a specific cluster, with columns 'Gene' indicating gene names.
surdata	A data frame containing survival information for the samples. The first column must be sample IDs.
time	A character string specifying the column name in 'surdata' representing survival time (default: "time").
event	A character string specifying the column name in 'surdata' representing the event status (default: "event").
methods	A character vector specifying which classifiers to use in the ensemble. If NULL, all available methods will be used (default: NULL).
sur.trend.rank	A character vector specifying the desired order of survival trends (e.g., c("C2", "C3", "C1")) to filter the models (default: NULL). Note: the order should be from risk to protective.
cutoff.P	Numeric value for the p-value cutoff for survival analysis (default: 0.05).

Details

This function runs an ensemble of classifiers, checks the consistency among classifiers, and optionally performs survival analysis to filter models based on trends in clinical outcomes.

Value

A list containing the ensemble prediction results and optional survival analysis.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
# Example usage:
data.test <- matrix(rnorm(1000), nrow = 100, ncol = 10)
data.train <- matrix(rnorm(1000), nrow = 100, ncol = 10)
cluster.data <- data.frame(Sample = paste0("Sample", 1:10), Cluster = rep(1:2, each = 5))
cluster.markers <- setNames(lapply(unique(cluster.data$Cluster), function(c) data.frame(Gene = paste0("Gene", sam
surdata <- data.frame(ID = paste0("Sample", 1:10), time = runif(10, 1, 1000), event = sample(0:1, 10, replace = TRUE
result <- RunEnsemble(data.test, data.train, cluster.data, cluster.markers, surdata, time = "time", event = "event
```

RunGSVA *Generate Single-Sample Gene-Set Enrichment Score*

Description

This function estimates gene-set enrichment scores across all samples using various methods.

Usage

```
RunGSVA(
  exp,
  gene.list,
  min.size = 3,
  max.size = 1000,
  method = "ssgsea",
  ssgsea.normalize = TRUE,
  ssgsea.alpha = 0.25,
  gsva.kcdf = "Gaussian",
  gsva.tau = 1,
  gsva.maxDiff = TRUE,
  gsva.absRanking = FALSE,
  verbose = TRUE,
  nCores = parallel::detectCores() - 3
)
```

Arguments

exp	Numeric matrix containing the expression data or gene expression signatures, with samples in columns and genes in rows.
gene.list	Gene sets provided either as a list object or as a GeneSetCollection object.
min.size	Minimum size of the gene sets to be considered in the analysis. Default is 3.
max.size	Maximum size of the gene sets to be considered in the analysis. Default is 1000.
method	Method to employ in the estimation of gene-set enrichment scores per sample. Options are "gsva" (default), "ssgsea", "zscore", or "plage".

`ssgsea.normalize`
: Logical vector of length 1; if TRUE runs the ssGSEA method from Barbie et al. (2009) normalizing the scores by the absolute difference between the minimum and the maximum, as described in their paper. Otherwise this last normalization step is skipped.

`ssgsea.alpha`
: Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the ssGSEA (Barbie et al., 2009) method. The default value is 0.25 as described in the paper.

`gsva.kcdf`
: Character vector of length 1 denoting the kernel to use during the non-parametric estimation of the cumulative distribution function of expression levels across samples. By default, kcdf="Gaussian" which is suitable when input expression values are continuous, such as microarray fluorescent units in logarithmic scale, RNA-seq log-CPMs, log-RPKMs or log-TPMs. When input expression values are integer counts, such as those derived from RNA-seq experiments, then this argument should be set to kcdf="Poisson".

`gsva.tau`
: Numeric vector of length 1. The exponent defining the weight of the tail in the random walk performed by the GSVA (Hänzelmann et al., 2013) method. The default value is 1 as described in the paper.

`gsva.maxDiff`
: Logical vector of length 1 which offers two approaches to calculate the enrichment statistic (ES) from the KS random walk statistic. FALSE: ES is calculated as the maximum distance of the random walk from 0. TRUE (the default): ES is calculated as the magnitude difference between the largest positive and negative random walk deviations.

`gsva.absRanking`
: Logical vector of length 1 used only when maxDiff=TRUE. When absRanking=FALSE (default) a modified Kuiper statistic is used to calculate enrichment scores, taking the magnitude difference between the largest positive and negative random walk deviations. When absRanking=TRUE the original Kuiper statistic that sums the largest positive and negative random walk deviations, is used. In this latter case, gene sets with genes enriched on either extreme (high or low) will be regarded as 'highly' activated.

`verbose`
: Logical indicating whether to print progress messages. Default is TRUE.

`nCores`
: The number of cores to use for parallel computation. Default is 'parallel::detectCores() - 2', which detects the number of cores available on the system and reserves 2 cores for other tasks.

Details

This function supports multiple methods for estimating gene-set enrichment scores, including ssGSEA, GSVA, zscore, and plage. The scores are calculated for each gene set across all samples. The 'ssGSES' function is flexible and allows for customization of the minimum and maximum size of gene sets considered in the analysis. By providing different methods, the function can adapt to various types of gene-set enrichment analysis, each having its own strengths and suitable applications.

- `"gsva"`: Gene Set Variation Analysis, suitable for detecting subtle changes in pathway activity.

- "ssgsea": Single-Sample Gene Set Enrichment Analysis, useful for individual sample analysis.
- "zscore": Z-score transformation, a simpler approach to standardize expression values.
- "plage": Pathway Level Analysis of Gene Expression, which focuses on correlating pathway components.

Value

A gene-set by sample matrix of gene-set enrichment scores.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

RuniClusterBayes *Run Bayesian Integrative Clustering (iClusterBayes)*

Description

Performs multi-omics clustering using iClusterBayes algorithm.

Usage

```
RuniClusterBayes(data, N.clust, method = "fast", burnin = 500, nsample = 1000)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
method	Character. "fast" or "full" (default: "fast").
burnin	Integer. Burn-in iterations for full method (default: 500).
nsample	Integer. Sampling iterations for full method (default: 1000).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Mo Q, et al. Biostatistics. 2018.

RunIntegration *Run Multi-Omics Integration Clustering*

Description

This function runs multi-omics integration analysis using a specified clustering algorithm. Users can choose from a variety of algorithms to perform data integration on multiple modalities.

Usage

```
RunIntegration(data, algorithm, N.clust, ...)

RunIF(data, algorithm, N.clust, ...)
```

Arguments

data	A list of matrices where each element represents a different modality (e.g., RNA, protein, methylation). Each matrix should have rows as features and columns as samples.
algorithm	Character. The integration algorithm to use. Options include "cpca", "iclusterbayes", "intnmf", "lracluster", "mcia", "nemo", "pinsplus", "rgcca", "sgcca", "snf", "cimlr", "bcc".
N.clust	Integer. Number of clusters to create (recommended).
...	Additional algorithm-specific arguments passed to the underlying functions.

Details

This function provides a unified interface to multiple multi-omics integration algorithms. Each algorithm has its own characteristics:

- SNF: Similarity Network Fusion
- CPCA: Consensus PCA
- iClusterBayes: Bayesian integrative clustering
- IntNMF: Integrative Non-negative Matrix Factorization
- LRAcluster: Low-Rank Approximation clustering
- MCIA: Multiple Co-inertia Analysis
- NEMO: Neighborhood based multi-omics clustering
- PINSPlus: Perturbation clustering for data integration
- RGCCA: Regularized Generalized CCA
- SGCCA: Sparse Generalized CCA
- CIMLR: Cancer Integration via Multi-kernel Learning
- BCC: Bayesian Consensus Clustering

Value

A data frame with clustering results based on the selected algorithm.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
## Not run:
# Create example data
data1 <- matrix(rnorm(5000), nrow = 50, ncol = 100)
data2 <- matrix(rnorm(5000), nrow = 50, ncol = 100)
colnames(data1) <- colnames(data2) <- paste0("Sample", 1:100)
data_list <- list(data1, data2)

# Run integration clustering using SNF
result <- RunIntegration(data = data_list, algorithm = "snf", N.clust = 3)

## End(Not run)
```

RunIntNMF *Run Integrative Non-negative Matrix Factorization (IntNMF)*

Description

Performs multi-omics clustering using IntNMF algorithm.

Usage

```
RunIntNMF(data, N.clust, maxiter = 200, n.ini = 30, wt = NULL)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
maxiter	Integer. Maximum iterations (default: 200).
n.ini	Integer. Number of initializations (default: 30).
wt	Numeric vector. Weights for each data type.

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Chalise P, Fridley BL. PLoS One. 2017.

`RunLRAcluster` *Run Low-Rank Approximation Clustering (LRAcluster)*

Description

Performs multi-omics clustering using LRAcluster algorithm.

Usage

```
RunLRAcluster(data, N.clust, data.types = NULL, cluster.algorithm = "ward.D2")
```

Arguments

`data`	A list of matrices (features x samples).
`N.clust`	Integer. Number of clusters.
`data.types`	Character vector. Data types ("binary", "gaussian", "poisson").
`cluster.algorithm`	Character. Clustering method (default: "ward.D2").

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Wu D, et al. BMC Genomics. 2015.

RunMCIA *Run Multiple Co-Inertia Analysis (MCIA)*

Description

Performs multi-omics clustering using MCIA algorithm.

Usage

```
RunMCIA(data, N.clust, ncomp = NULL, clustering.algorithm = "ward.D2")
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
ncomp	Integer. Number of components (default: auto).
clustering.algorithm	Character. Clustering method (default: "ward.D2").

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Meng C, et al. BMC Bioinformatics. 2014.

RunMOFS *Run MultiModality Fusion Subtyping (MOFS) for Multi-Modality Data Integration*

Description

This function performs MultiModality Fusion Subtyping (MOFS) analysis by utilizing multiple clustering algorithms for multi-modality data integration. The user can flexibly select the desired clustering algorithms and adjust relevant parameters.

Usage

```
RunMOFS(
  data,
  methods,
  max.clusters = 6,
  optimal.clusters = 3,
  linkage.method = "ward.D2",
  clustering.algorithm = "hc",
  distance.metric = "euclidean",
  resampling.iterations = 10000,
  resample.proportion = 0.7,
  silhouette.cutoff = 0.4,
  ...
)
```

Arguments

`data`	A list of matrices where each element represents a different modality (e.g., RNA, protein, methylation). Each matrix should have rows as features and columns as samples.
`methods`	Character vector. The clustering algorithms to use. Options are: "CPCA", "iClusterBayes", "IntNMF", "LRAcluster", "MCIA", "NEMO", "PINSPlus", "RGCCA", "SGCCA", "SNF", "CIMLR", "BCC". At least two methods must be specified.
`max.clusters`	Integer. The maximum number of clusters to evaluate during consensus clustering analysis (default: 6).
`optimal.clusters`	Integer. The optimal number of clusters to select from the consensus clustering analysis (default: 3).
`linkage.method`	Character. The linkage method to use for hierarchical clustering (default: "ward.D2").
`clustering.algorithm`	Character. The clustering algorithm to use during consensus clustering (default: 'hc').
`distance.metric`	Character. The distance metric to use for clustering (default: "euclidean").
`resampling.iterations`	Integer. The number of resampling iterations for consensus clustering (default: 10000).
`resample.proportion`	Numeric. The proportion of items to resample in each iteration for consensus clustering (default: 0.7).
`silhouette.cutoff`	Numeric. Silhouette coefficient cutoff value for selecting core set samples (default: 0.4).
`...`	Additional parameters specific to the chosen clustering algorithms.

Details

The function performs MultiModality Fusion Subtyping (MOFS) by running multiple clustering algorithms on the input multi-modality data. The results of each clustering algorithm are stored for further analysis, including binary cluster assignments, Jaccard distance calculation, consensus clustering analysis, Calinski-Harabasz index calculation, silhouette analysis, and PCA.

The steps involved are: 1. Running the specified clustering algorithms on the input data. 2. Extracting binary cluster assignments from the clustering results. 3. Calculating Jaccard distance between clusters. 4. Performing consensus clustering analysis to identify stable clusters. 5. Calculating the Calinski-Harabasz index to assess clustering quality. 6. Performing silhouette analysis to evaluate cluster cohesion and separation. 7. Identifying a core set of samples based on the silhouette coefficient cutoff. 8. Performing PCA on the core set for dimensionality reduction and visualization.

Value

A list containing the results for each specified clustering algorithm, as well as the results of further analysis including consensus clustering, silhouette scores, core set identification, and PCA.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

Examples

```
# Example usage:
data1 <- matrix(rnorm(10000), nrow = 100, ncol = 100)
data2 <- matrix(rnorm(10000), nrow = 100, ncol = 100)
colnames(data1) <- colnames(data2) <- paste0("Sample", 1:100)
data_list <- list(data1, data2)

# Run MultiModality Fusion Subtyping using CPCA and CIMLR
result <- RunMOFS(data = data_list, methods = c("CPCA", "CIMLR"), max.clusters = 6, optimal.clusters = 3, linkage.m
```

RunNEMO *Run Neighborhood-based Multi-Omics clustering (NEMO)*

Description

Performs multi-omics clustering using NEMO algorithm.

Usage

```
RunNEMO(data, N.clust = NULL, num.neighbors = NA)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters (NULL for automatic).
num.neighbors	Number of neighbors (NA for automatic).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Rappoport N, Shamir R. Bioinformatics. 2019.

`RunPCA` *Run Principal Component Analysis*

Description

Performs PCA on the given data using base R.

Usage

```
RunPCA(data, scale = TRUE, center = TRUE, ncomp = NULL)
```

Arguments

`data`	A data matrix or distance matrix.
`scale`	Logical. Whether to scale variables (default: TRUE).
`center`	Logical. Whether to center variables (default: TRUE).
`ncomp`	Number of components to return (default: all).

Value

A list with class "prcomp" containing PCA results.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

RunPINSPlus *Run Perturbation Clustering (PINSPlus)*

Description

Performs multi-omics clustering using PINSPlus algorithm.

Usage

```
RunPINSPlus(data, N.clust = NULL, kMin = 2, kMax = 5)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters (NULL for automatic).
kMin	Integer. Minimum clusters (default: 2).
kMax	Integer. Maximum clusters (default: 5).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Nguyen T, et al. Genome Research. 2017;27(12):2025-2039.

RunRGCCA *Run Regularized Generalized Canonical Correlation Analysis (RGCCA)*

Description

Performs multi-omics clustering using RGCCA algorithm.

Usage

```
RunRGCCA(
  data,
  N.clust,
  ncomp = NULL,
  tau = NULL,
  scheme = "centroid",
  clustering.algorithm = "ward.D2"
)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
ncomp	Integer vector. Number of components per block.
tau	Numeric vector. Shrinkage parameters.
scheme	Character. Scheme type ("centroid", "factorial", "horst").
clustering.algorithm	Character. Clustering method (default: "ward.D2").

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Tenenhaus A, Tenenhaus M. Psychometrika. 2011;76(2):257-284.

RunSGCCA *Run Sparse Generalized Canonical Correlation Analysis (SGCCA)*

Description

Performs multi-omics clustering using SGCCA algorithm.

Usage

```
RunSGCCA(
  data,
  N.clust,
  ncomp = NULL,
  c1 = NULL,
  scheme = "centroid",
  clustering.algorithm = "ward.D2"
)
```

Arguments

data	A list of matrices (features x samples).
N.clust	Integer. Number of clusters.
ncomp	Integer vector. Number of components per block.
c1	Numeric vector. L1 penalty parameters (0-1).
scheme	Character. Scheme type (default: "centroid").
clustering.algorithm	Character. Clustering method (default: "ward.D2").

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Tenenhaus et al. Biostatistics. 2013.

RunSNF *Run Similarity Network Fusion (SNF) for Multi-Modality Data Integration*

Description

This function performs clustering analysis using Similarity Network Fusion (SNF), which integrates multiple data types to provide a unified clustering result.

Usage

```
RunSNF(
  data,
  N.clust = NULL,
  num.neighbors = 20,
  variance = 0.5,
  num.iterations = 20
)
```

Arguments

data	A list of matrices where each element represents a different modality. Each matrix should have rows as features and columns as samples.
N.clust	Integer. Number of clusters for spectral clustering.
num.neighbors	Integer. Number of nearest neighbors (default: 20).
variance	Numeric. Variance for local model (default: 0.5).
num.iterations	Integer. Number of SNF iterations (default: 20).

Value

A data frame with Sample, Cluster, and Cluster2 columns.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Wang B, et al. Nat Methods. 2014;11(3):333-337.

SD.df *Calculate Standard Deviation for a Data Frame*

Description

This function calculates the standard deviation (SD) for each column in a data frame.

Usage

```
SD.df(df)
```

Arguments

df	A data frame with row samples and column features.

Value

A sorted vector of SD values for each column in the data frame, in decreasing order.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

`Select.Features` *Select Hypervariable Features*

Description

This function selects hypervariable features from a data frame based on specified variance calculation methods.

Usage

```
Select.Features(
  data,
  method = "mad",
  top.percent = NULL,
  top.number = 1000,
  custom.features = NULL
)
```

Arguments

`data`	A data frame with row features and column samples.
`method`	Method of calculating variance ("sd", "mad", "cv").
`top.percent`	The top percent of hypervariable features based on variance.
`top.number`	The top number of hypervariable features based on variance.
`custom.features`	A vector of custom features to select.

Value

A vector of selected hypervariable features.

Author(s)

Zaoqu Liu; E-mail: liuzaoqu@163.com

Examples

```
## Not run:
df <- data.frame(matrix(rnorm(1000), nrow = 100, ncol = 10))
selected_features <- Select.Features(df, method = "mad", top.percent = 10)

## End(Not run)
```

sgcca *Sparse Generalized Canonical Correlation Analysis*

Description

RGCCA with L1 sparsity for variable selection.

Usage

```
sgcca(
  A,
  C = 1 - diag(length(A)),
  c1 = rep(1, length(A)),
  ncomp = rep(1, length(A)),
  scheme = "centroid",
  scale = TRUE,
  init = "svd",
  bias = TRUE,
  tol = .Machine$double.eps,
  verbose = FALSE
)
```

Arguments

A	List of data blocks.
C	Design matrix.
c1	L1 penalty (between 0 and 1).
ncomp	Number of components per block.
scheme	Scheme type.
scale	Scale blocks.
init	Initialization method.
bias	Biased estimator.
tol	Tolerance.
verbose	Print progress.

Value

List with Y, a, astar, C, c1, scheme, ncomp, crit, AVE.

snf_fuse *Similarity Network Fusion*

Description

Fuses multiple affinity matrices into a unified similarity network.

Usage

```
snf_fuse(Wall, K = 20, t = 20)
```

Arguments

Wall	List of affinity matrices (one per data type).
K	Number of neighbors for KNN step (default: 20).
t	Number of iterations for diffusion process (default: 20).

Value

Unified similarity matrix.

spectral_clustering *Spectral Clustering*

Description

Performs spectral clustering on an affinity matrix.

Usage

```
spectral_clustering(affinity, K, type = 3)
```

Arguments

affinity	Affinity matrix (N x N).
K	Number of clusters.
type	Type of Laplacian normalization (1, 2, or 3; default: 3).

Value

Vector of cluster labels (1 to K).

ssMwwGST *Single-Sample Pathway Activity Analysis Using MWW-GST*

Description

Performs single-sample pathway activity analysis using Mann-Whitney-Wilcoxon Gene Set Test (MWW-GST) method.

Usage

```
ssMwwGST(geData, geneSet, nCores = 1, minLenGeneSet = 15)
```

Arguments

geData	A numeric matrix of gene expression (genes x samples).
geneSet	A list of gene sets (vectors of gene names).
nCores	Integer. Number of cores for parallel processing (default: 1).
minLenGeneSet	Minimum gene set size (default: 15).

Value

A list containing NES, pValue, and FDR matrices.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Frattini V, et al. Nature. 2018;553(7687):222-227.

stop_parallel *Stop Parallel Backend*

Description

Resets to sequential processing.

Usage

```
stop_parallel()
```

subtyping_omics_data *Subtyping Multi-Omics Data with PINSPlus*

Description

Performs multi-omic clustering using perturbation clustering.

Usage

```
subtyping_omics_data(
  data_list,
  kMin = 2,
  kMax = 5,
  k = NULL,
  agreement_cutoff = 0.5,
  verbose = FALSE
)
```

Arguments

data_list	List of data matrices (samples x features).
kMin	Minimum clusters (default: 2).
kMax	Maximum clusters (default: 5).
k	Fixed number of clusters (optional).
agreement_cutoff	Agreement threshold (default: 0.5).
verbose	Print progress.

Value

List with cluster1, cluster2 and data type results.

WangGBM *Perform ssGSEA-based Subtyping for GBM Samples (Wang et al. 2017)*

Description

This function performs single-sample Gene Set Enrichment Analysis (ssGSEA) for Glioblastoma Multiforme (GBM) data based on the Wang et al. 2017 classification system. It predicts sample subtypes (Classical, Mesenchymal, Proneural) based on enrichment scores using established marker gene sets.

Usage

```
WangGBM(
  data.test,
  dir.file = ".",
  gct.filename = "data.gct",
  number.perms = 100,
  tolerate.mixed = FALSE,
  method = c("internal", "GSVA", "external")
)
```

Arguments

data.test	A matrix or data frame representing the input expression data, where rows are genes and columns are samples.
dir.file	Character. Directory for saving the output files (default: '.'). Set to NULL to use a temporary directory.
gct.filename	Character. The filename for the generated GCT file (default: 'data.gct').
number.perms	Integer. Number of permutations for ssGSEA analysis (default: 100).
tolerate.mixed	Logical. Whether to allow "Mixed" predictions when multiple gene sets have the same minimum p-value (default: FALSE).
method	Character. The ssGSEA implementation to use: "internal" (built-in), "GSVA" (requires GSVA package), or "external" (requires ssgsea.GBM.classification package from GitHub). Default: "internal".

Details

The function uses the Wang et al. 2017 GBM subtyping system which classifies samples into three subtypes:

- Classical (CL)
- Mesenchymal (MES)
- Proneural (PN)

For the "external" method, the ssgsea.GBM.classification package is required: `devtools::install_github("Zaoqu-Liu/s`

Value

A data frame with the following columns:

- `ID`: Sample identifiers.
- `Predict`: Predicted subtype for each sample.
- Columns with `_pval`: P-values for each subtype.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Wang Q, Hu B, Hu X, Kim H, Squatrito M, Scarpace L, et al. Tumor Evolution of Glioma-Intrinsic Gene Expression Subtypes Associates with Immunological Changes in the Microenvironment. Cancer Cell. July 2017;32(1):42-56.e6.

Examples

```
## Not run:
# Simulated expression data
data.test <- matrix(rnorm(10000), nrow = 100, ncol = 100)
rownames(data.test) <- paste0("Gene", 1:100)
colnames(data.test) <- paste0("Sample", 1:100)

# Run GBM ssGSEA-based subtyping
result <- WangGBM(
  data.test = data.test,
  number.perms = 50,
  tolerate.mixed = TRUE
)
print(result)

## End(Not run)
```

WuGBM *Perform ssGSEA-based Subtyping Using Wu et al. 2024 Markers*

Description

This function performs single-sample Gene Set Enrichment Analysis (ssGSEA) to classify GBM samples into subtypes based on Wu et al. 2024 molecular markers, including Proneural (PN), Mesenchymal (MES), and Oxidative Phosphorylation (OXPHOS).

Usage

```
WuGBM(
  data.test,
  dir.file = ".",
  gct.filename = "data.gct",
  number.perms = 100,
  tolerate.mixed = FALSE,
  method = c("internal", "GSVA", "external")
)
```

Arguments

data.test	A matrix or data frame representing the input expression data, where rows are genes and columns are samples.

dir.file	Character. Directory for saving the output files (default: '.'). Set to NULL to use a temporary directory.
gct.filename	Character. The filename for the generated GCT file (default: 'data.gct').
number.perms	Integer. Number of permutations for ssGSEA analysis (default: 100).
tolerate.mixed	Logical. Whether to allow "Mixed" predictions when multiple gene sets have the same minimum p-value (default: FALSE).
method	Character. The ssGSEA implementation to use: "internal" (built-in), "GSVA" (requires GSVA package), or "external". Default: "internal".

Details

The function uses predefined marker gene sets for GBM subtypes from Wu et al. 2024:

- PN: Proneural subtype markers.
- OXPHOS: Oxidative phosphorylation subtype markers.
- MES: Mesenchymal subtype markers.

Value

A data frame with the following columns:

- ID: Sample identifiers.
- Predict: Predicted subtype for each sample.
- Columns with _pval: P-values for each subtype.

Author(s)

Zaoqu Liu; Email: liuzaoqu@163.com

References

Wu M, Wang T, Ji N, Lu T, Yuan R, Wu L, et al. Multi-omics and pharmacological characterization of patient-derived glioma cell lines. Nat Commun. 2024;15:6740. doi:10.1038/s41467-024-51214-y.

Examples

```
## Not run:
# Simulated expression data
data.test <- matrix(rnorm(10000), nrow = 100, ncol = 100)
rownames(data.test) <- paste0("Gene", 1:100)
colnames(data.test) <- paste0("Sample", 1:100)

# Run WuGBM subtyping
result <- WuGBM(
  data.test = data.test,
  number.perms = 50,
  tolerate.mixed = TRUE
)
```

```
print(result)

## End(Not run)
```

Index

∗ **datasets**
    algo-nemo, 7
    gene_sets, 51
.NEMO_NEIGHBORS_RATIO *(*algo-nemo*)*, 7

algo-factor, 5
algo-iclusterbayes, 6
algo-nemo, 7
algo-snf, 7
align_samples, 8

bcc_cluster, 8
bcc_cluster_fast, 9

calc_chi, 10
calc_pac, 10
CalCHI, 11
CalPAC, 11
check_sample_alignment, 12
cimlr_cluster, 13
cimlr_feature_ranking, 13
Classifier.Adaboost, 14
Classifier.DT, 15
Classifier.Enet, 17
Classifier.Enrichment, 19
Classifier.GBDT, 20
Classifier.kNN, 22
Classifier.LASSO, 24
Classifier.LDA, 25
Classifier.NBayes, 27
Classifier.NNet, 29
Classifier.PCA, 30
Classifier.RF, 32
Classifier.Ridge, 34
Classifier.ssGSEA, 35
Classifier.StepLR, 37
Classifier.SVD, 39
Classifier.SVM, 41
Classifier.XGBoost, 42
compare_clusterings, 44
compute_umap, 44
consensus_cluster, 45
correct_batch, 46
cpca, 46
CV, 47
CV.df, 47

FeatureSelectionWithBootstrap, 48
filter_by_mad, 49
filter_low_variance, 49
Find.OptClusterFeatures, 50

gene_sets, 51
get.binary.clusters, 51
get.class, 52
get.Jaccard.Distance, 52
get_consensus_class, 53

handle_missing, 54

icluster_bayes *(*algo-iclusterbayes*)*, 6
icluster_bayes_fast, 54
init, 55
intnmf_cluster, 55
intnmf_opt_k, 56

list_clustering_algorithms
        *(*run-clustering*)*, 72
lracluster, 57

MAD.df, 57
mcia, 58
Mean.df, 58
Median.df, 59
minmax, 59
minmax.df, 60
multi_view_factor_analysis
        *(*algo-factor*)*, 5

nemo_affinity_graph, 60
nemo_clustering, 61

nemo_num_clusters, 61
normalize_omics *(*preprocessing*)*, 70

parallel, 62
parallel_bootstrap_features, 62
parallel_consensus_cluster, 63
PathDEA, 64
perturbation_clustering, 65
plot_algorithm_comparison, 66
plot_cluster_quality, 67
plot_consensus_heatmap, 67
plot_silhouette, 68
plot_survival, 69
plot_umap, 69
preprocessing, 70

qc_summary, 71

rgcca, 71
run-clustering, 72
run_bcc, 73
run_cimlr, 73
run_cpca, 74
run_iclusterbayes, 74
run_integration, 75
run_intnmf, 76
run_late_fusion, 76
run_lracluster, 77
run_mcia, 77
run_mofa, 78
run_multiple_algorithms, 78
run_nemo, 79
run_parallel_algorithms, 79
run_pinsplus, 80
run_rgcca, 80
run_sgcca, 81
run_snf, 82
run_wsnf, 82
RunBCC, 83
RunCC, 84
RunCIMLR, 85
RunClassifier, 85
RunCOCA, 87
RunCPCA, 88
RunEnsemble, 88
RunGSVA, 90
RuniClusterBayes, 92
RunIF *(*RunIntegration*)*, 93
RunIntegration, 93
RunIntNMF, 94
RunLRAcluster, 95
RunMCIA, 96
RunMOFS, 96
RunNEMO, 98
RunPCA, 99
RunPINSPlus, 100
RunRGCCA, 100
RunSGCCA, 101
RunSNF, 102

SD.df, 103
Select.Features, 104
setup_parallel *(*parallel*)*, 62
sgcca, 105
snf_affinity_matrix *(*algo-snf*)*, 7
snf_fuse, 106
spectral_clustering, 106
ssMwwGST, 107
stop_parallel, 107
subtyping_omics_data, 108

WangGBM, 108
WuGBM, 110