

# Package ‘iModMix’

May 19, 2026

**Title** Integrative Modules for Multi-Omics Data

**Version** 1.3.0

**Description** The iModMix network-based method offers an integrated framework for analyzing multi-omics data, including metabolomics, proteomics, and transcriptomics data, enabling the exploration of intricate molecular associations within heterogeneous biological systems.

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**biocViews** Software, Network, Clustering, Visualization,  
Transcriptomics, Proteomics, Metabolomics, GeneExpression,  
PrincipalComponent

**URL** <https://github.com/biodatalab/iModMix>

**BugReports** <https://github.com/biodatalab/iModMix/issues>

**Depends** R (>= 4.5.0)

**Imports** config (>= 0.3.2), golem (>= 0.4.1), shiny (>= 1.7.5),  
ComplexHeatmap, DT, RColorBrewer, WGCNA, corrplot, cowplot,  
dynamicTreeCut, ggplot2, glassoFast, impute, purrr, stringr,  
tidyr, visNetwork, shinyBS, httr, dplyr, stats, iModMixData,  
SummarizedExperiment, ExperimentHub (>= 2.99.0)

**Suggests** testthat (>= 3.0.0), ggfortify, shinyWidgets, pROC, tuneR,  
knitr, curl, readxl, reshape2, vroom, here, enrichR, rmarkdown

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/iModMix>

**git\_branch** devel

**git\_last\_commit** 646836c

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.24

**Date/Publication** 2026-05-18

**Author** Isis Narvaez-Bandera [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-7320-618X>>)

**Maintainer** Isis Narvaez-Bandera <isis.narvaez@upr.edu>

## Contents

fctAssignmentGenesEnrichr . . . . .	2
fctClusterAssignments . . . . .	3
fctEigengenes . . . . .	4
fctFeaturesAnnotCorrelation . . . . .	5
fctHierarchicalCluster . . . . .	6
fctiModMix2SE . . . . .	7
fctLoadData . . . . .	8
fctModulesCorrelation . . . . .	8
fctPartialCors . . . . .	9
fctPerformClassification . . . . .	10
run_app . . . . .	11
<b>Index</b>	<b>13</b>

---

fctAssignmentGenesEnrichr  
*fctAssignmentGenesEnrichr*

---

## Description

Enrichr terms for genes (Proteins and transcriptomics)

## Usage

```
fctAssignmentGenesEnrichr(
  clusterAssignmentsProtGenes,
  database = "GO_Biological_Process_2023"
)
```

## Arguments

clusterAssignmentsProtGenes  
 (data frame containing cluster\_assignments and HMDB/Symbol)

database        A list with all the available databases from enrichr

## Value

result\_list data frame containing cluster\_assignments and Enrichr terms

## Examples

```
if (interactive()) {
  # Simulated cluster assignments with gene symbols
  cluster_assignments <- data.frame(
    feature = c("F1", "F2", "F3", "F4"),
    cluster = c("cluster_1", "cluster_1", "cluster_2", "cluster_2"),
    col = c("#FF0000", "#FF0000", "#00FF00", "#00FF00"),
    feature_name = c("TP53", "BRCA1", "EGFR", "MYC"),
    stringsAsFactors = FALSE
  )
}
```

```
# Run enrichment analysis using Enrichr (requires internet connection)
enriched_results <- fctAssignmentGenesEnrichr(
  clusterAssignmentsProtGenes = cluster_assignments,
  database = "GO_Biological_Process_2023"
)

# View results
head(enriched_results)
}
```

---

fctClusterAssignments *fctClusterAssignments*

---

## Description

Cluster assignments with annotation.

## Usage

```
fctClusterAssignments(cluster, phenoData = NULL, selectedColumns = NULL)
```

## Arguments

cluster	A data frame containing cluster assignments.
phenoData	A data frame with the annotation names of the Dataset. Should have a column called <code>Feature_ID</code> .
selectedColumns	Columns selected for the user to merge in the cluster assignments table

## Value

A data frame containing cluster assignments and selected columns by user.

## Examples

```
# Simulated cluster assignment data
cluster_df <- data.frame(
  feature = paste0("F", 1:5),
  cluster = paste0("cluster_", 1:5),
  col = RColorBrewer::brewer.pal(5, "Set1"),
  stringsAsFactors = FALSE
)

# Simulated annotation data
pheno_df <- data.frame(
  Feature_ID = paste0("F", 1:5),
  Symbol = c("GeneA", "GeneB", "GeneC", "GeneD", "GeneE"),
  Description = paste("Description", 1:5),
  stringsAsFactors = FALSE
)

# Run without selected columns
```

```
fctClusterAssignments(cluster_df, phenoData = pheno_df)

# Run with selected columns
fctClusterAssignments(cluster_df, phenoData = pheno_df, selectedColumns = "Description")
```

---

fctEigengenes	<i>fctEigengenes</i>
---------------	----------------------

---

## Description

Calculates module eigengenes using the WGCNA package.

## Usage

```
fctEigengenes(loadData = loadData, clusterAssignments = clusterAssignments)
```

## Arguments

`loadData`            A preprocessed data matrix resulting from "load data" function  
`clusterAssignments`            A vector of cluster assignments for each feature.

## Value

A list containing:

`module_eigenmetab_List_Me`            The full result from WGCNA::moduleEigengenes.  
`module_eigenmetab_Me`            The matrix of module eigengenes.  
`feature_mat_t`            The filtered and scaled feature matrix.

## Examples

```
# Simulate a small expression matrix
set.seed(123)
mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
colnames(mat) <- paste0("Gene", 1:10)
rownames(mat) <- paste0("Sample", 1:10)

# Simulate cluster assignments
clusters <- rep(1:2, each = 5)

# Run Eigengenes function
result <- fctEigengenes(loadData = mat, clusterAssignments = clusters)

# View eigengene matrix
head(result$module_eigenmetab_Me)
```

---

```
fctFeaturesAnnotCorrelation
      fctFeaturesAnnotCorrelation
```

---

## Description

Calculate correlation between the features of top correlated modules.

## Usage

```
fctFeaturesAnnotCorrelation(
  corDataiDataj,
  clusterAssignmentsD1,
  clusterAssignmentsD2,
  loadData1 = loadData1,
  loadData2 = loadData2,
  topn = 5
)
```

## Arguments

`corDataiDataj` A data frame with the first principal component of each protein cluster and their correlations.

`clusterAssignmentsD1` A data frame containing cluster assignments for metabolites.

`clusterAssignmentsD2` A data frame containing cluster assignments and Enrichr terms for proteins.

`loadData1` A preprocessed data matrix resulting from "load data" function from Data1.

`loadData2` A preprocessed data matrix resulting from "load data" function from Data2.

`topn` The number of top correlations to select.

## Value

A list containing important features and correlation matrices.

## Examples

```
# Simulated correlation data
corDataiDataj <- data.frame(
  from = c("blue", "green"),
  to = c("red", "yellow"),
  value = c(0.9, 0.85),
  stringsAsFactors = FALSE
)

# Simulated cluster assignments
clusterAssignmentsD1 <- data.frame(
  feature = c("M1", "M2"),
  col = c("blue", "green"),
  stringsAsFactors = FALSE
)
```

```

clusterAssignmentsD2 <- data.frame(
  feature = c("P1", "P2"),
  col = c("red", "yellow"),
  stringsAsFactors = FALSE
)

# Simulated expression matrices
loadData1 <- matrix(rnorm(20), nrow = 5, ncol = 4)
colnames(loadData1) <- c("M1", "M2", "X1", "X2")
rownames(loadData1) <- paste0("Sample", 1:5)

loadData2 <- matrix(rnorm(20), nrow = 5, ncol = 4)
colnames(loadData2) <- c("P1", "P2", "Y1", "Y2")
rownames(loadData2) <- paste0("Sample", 1:5)

# Run the function
result <- fctFeaturesAnnotCorrelation(
  corDataiDataj = corDataiDataj,
  clusterAssignmentsD1 = clusterAssignmentsD1,
  clusterAssignmentsD2 = clusterAssignmentsD2,
  loadData1 = loadData1,
  loadData2 = loadData2,
  topn = 2
)

# View top correlations
result$Top_correlations

```

---

fctHierarchicalCluster

*fctHierarchicalCluster*

---

## Description

Perform hierarchical clustering of a partial correlation matrix. By default uses the topological overlap measure (TOM)

## Usage

```
fctHierarchicalCluster(parcorMat, tom = TRUE, minModuleSize = 10)
```

## Arguments

parcorMat	A partial correlation matrix from <code>glassoFast\$wi</code> : Estimated inverse covariance matrix
tom	TRUE: topological overlap measure; FALSE: partial correlation matrix; default is TRUE.
minModuleSize	Smallest modules to be generated; default is 10

## Value

`result_list` List containing `TOM_diss` (TOM dissimilarity; NA if argument `TOM = FALSE`), `hclust-Tree` (hierarchical clustering object), `dynamicMods` (index of module membership for features), and `mod_count` (number of modules).

**Examples**

```
# Simulate a small partial correlation matrix
set.seed(123)
mat <- matrix(rnorm(100), nrow = 10)
colnames(mat) <- paste0("Gene", 1:10)
rownames(mat) <- paste0("Sample", 1:10)

# Compute covariance and partial correlation matrix
cov_mat <- cov(mat)
parcorMat <- glassoFast::glassoFast(cov_mat, rho = 0.25)$wi
diag(parcorMat) <- NA
colnames(parcorMat) <- rownames(parcorMat) <- paste0("Gene", 1:10)

# Run hierarchical clustering
result <- fctHierarchicalCluster(parcorMat, tom = TRUE, minModuleSize = 3)

# View module assignments
head(result$cluster_assignments)
```

---

fctiModMix2SE

*fctiModMix2SE*


---

**Description**

Converts expression data and corresponding sample metadata into a Bioconductor-compliant SummarizedExperiment object.

**Usage**

```
fctiModMix2SE(dataExp, metadata)
```

**Arguments**

dataExp	A data frame with metabolites (features) in rows and samples in columns. The first column should contain feature IDs (e.g., metabolite names).
metadata	A data frame containing sample metadata. Must include a column named "Sample" that matches the column names of dataExp (excluding the first column).

**Value**

A SummarizedExperiment object.

**Examples**

```
dataExp <- data.frame(Feature_ID = paste0("M", 1:5),
                     S1 = rnorm(5), S2 = rnorm(5))
metadata <- data.frame(Sample = c("S1", "S2"),
                      Group = c("A", "B"))
se <- fctiModMix2SE(dataExp, metadata)
se
```

fctLoadData

*fctLoadData*

---

**Description**

A function to load and preprocess the expression data.

**Usage**

```
fctLoadData(expressionMat = expressionMat)
```

**Arguments**

`expressionMat` A feature matrix (e.g., gene expression) with samples in rows and features (e.g., genes) in columns. Row names must be unique.

**Value**

A preprocessed data matrix.

**Examples**

```
# Simulated expression matrix with missing values
set.seed(123)
mat <- matrix(rnorm(1000), nrow = 10, ncol = 100)
colnames(mat) <- paste0("Gene", 1:100)
rownames(mat) <- paste0("Sample", 1:10)
mat[sample(length(mat), 50)] <- NA # introduce some NAs

# Add a Feature_ID column to mimic expected input
expressionMat <- cbind(Feature_ID = paste0("F", 1:10), mat)
expressionMat <- as.data.frame(expressionMat)

# Convert Feature_ID to character (if needed)
expressionMat$Feature_ID <- as.character(expressionMat$Feature_ID)

# Run the function
processed_data <- fctLoadData(expressionMat)
```

---

fctModulesCorrelation *fctModulesCorrelation*

---

**Description**

Calculates correlations between omic modules.

**Usage**

```
fctModulesCorrelation(eigengenesList, clusterList, threshold = 0.5)
```

**Arguments**

`eigengenesList` List with the first principal component of each cluster.  
`clusterList` list containing cluster\_assignments matrix.  
`threshold` A numeric value to filter correlations. Default is 0.5.

**Value**

Return the Top\_correlations between Prot and metab eigengenes, graph of red, correlation plot.

**Examples**

```
# Simulate eigengene matrices for two omics datasets
set.seed(123)
eig1 <- matrix(rnorm(50), nrow = 10, ncol = 5)
colnames(eig1) <- paste0("ME", 1:5)
eig2 <- matrix(rnorm(50), nrow = 10, ncol = 5)
colnames(eig2) <- paste0("ME", 1:5)

# Simulate cluster assignments for each omic
cluster1 <- data.frame(
  feature = paste0("F", 1:5),
  cluster = paste0("cluster_", 1:5),
  col = RColorBrewer::brewer.pal(5, "Set1"),
  stringsAsFactors = FALSE
)
cluster2 <- data.frame(
  feature = paste0("F", 6:10),
  cluster = paste0("cluster_", 6:10),
  col = RColorBrewer::brewer.pal(5, "Set2"),
  stringsAsFactors = FALSE
)

# Run the correlation function
result <- fctModulesCorrelation(
  eigengenesList = list(eig1, eig2),
  clusterList = list(cluster1, cluster2),
  threshold = 0.5
)

# View top correlations
head(result$Top_cor_Prot_metab)
```

---

fctPartialCors

*fctPartialCors*


---

**Description**

Calculates partial correlations based on graphical lasso (<https://www.rdocumentation.org/packages/glassoFast/versions/1>)

**Usage**

```
fctPartialCors(loadData, rho = 0.25)
```

**Arguments**

`loadData` A preprocessed data matrix resulting from "load data" function.

`rho` The regularization parameter for lasso. (a non-negative value or a p by p matrix of regularization parameters).

**Value**

`partial_cor_mat` A partial correlation matrix with NA's in the diagonal.

**Examples**

```
# Simulate a small expression matrix
set.seed(123)
mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
colnames(mat) <- paste0("Gene", 1:10)
rownames(mat) <- paste0("Sample", 1:10)

# Run partial correlation
partial_cor_matrix <- fctPartialCors(mat, rho = 0.25)

# View a portion of the result
partial_cor_matrix[1:5, 1:5]
```

---

fctPerformClassification  
*fctPerformClassification*

---

**Description**

Performs classification using different methods such as Welch's T-test, Random Forest, and K-Nearest Neighbors.

**Usage**

```
fctPerformClassification(
  eigengeneData,
  metadata,
  phenotypeVariable,
  significanceThreshold = 0.05
)
```

**Arguments**

`eigengeneData` A data frame of eigengenes organized with patient IDs in rows and variables in columns.

`metadata` A data frame containing metadata, with a column call "Sample" that matches patient IDs in `eigengeneData`.

`phenotypeVariable` The variable selected by the user in the Shiny app (response variable).

`significanceThreshold` A numeric value to filter p-value. Default is 00.5.

**Value**

A data frame with metrics such as AUC, Accuracy, and Error Rate for each binary classification.

**Examples**

```
# Simulate eigengene data
set.seed(123)
eigengeneData <- as.data.frame(matrix(rnorm(100), nrow = 10, ncol = 10))
colnames(eigengeneData) <- paste0("ME", 1:10)
rownames(eigengeneData) <- paste0("Sample", 1:10)
# Simulate metadata with a binary phenotype
metadata <- data.frame(
  Sample = paste0("Sample", 1:10),
  Phenotype = rep(c("A", "B"), each = 5),
  stringsAsFactors = FALSE
)
# Run classification
result <- fctPerformClassification(
  eigengeneData = eigengeneData,
  metadata = metadata,
  phenotypeVariable = "Phenotype",
  significanceThreshold = 0.05
)

# View results
head(result$result)
```

---

run\_app

*Run the Shiny Application*

---

**Description**

Run the Shiny Application

**Usage**

```
run_app(
  onStart = NULL,
  options = list(),
  enableBookmarking = NULL,
  uiPattern = "/",
  ...
)
```

**Arguments**

**onStart** A function that will be called before the app is actually run. This is only needed for shinyAppObj, since in the shinyAppDir case, a global.R file can be used for this purpose.

options	Named options that should be passed to the runApp call (these can be any of the following: "port", "launch.browser", "host", "quiet", "display.mode" and "test.mode"). You can also specify width and height parameters which provide a hint to the embedding environment about the ideal height/width for the app.
enableBookmarking	Can be one of "url", "server", or "disable". The default value, NULL, will respect the setting from any previous calls to <a href="#">enableBookmarking()</a> . See <a href="#">enableBookmarking()</a> for more information on bookmarking your app.
uiPattern	A regular expression that will be applied to each GET request to determine whether the ui should be used to handle the request. Note that the entire request path must match the regular expression in order for the match to be considered successful.
...	arguments to pass to golem_opts. See <code>'?golem::get_golem_options'</code> for more details.

**Value**

A Shiny application object that launches the iModMix interface.

# Index

`enableBookmarking()`, [12](#)

`fctAssignmentGenesEnrichr`, [2](#)

`fctClusterAssignments`, [3](#)

`fctEigengenes`, [4](#)

`fctFeaturesAnnotCorrelation`, [5](#)

`fctHierarchicalCluster`, [6](#)

`fctiModMix2SE`, [7](#)

`fctLoadData`, [8](#)

`fctModulesCorrelation`, [8](#)

`fctPartialCors`, [9](#)

`fctPerformClassification`, [10](#)

`run_app`, [11](#)