Package 'MSstatsBioNet'

April 6, 2026

Type Package

Title Network Analysis for MS-based Proteomics Experiments

Version 1.3.5

Description A set of tools for network analysis using mass spectrometry-based proteomics data and network databases. The package takes as input the output of MSstats differential abundance analysis and provides functions to perform enrichment analysis and visualization in the context of prior knowledge from past literature. Notably, this package integrates with INDRA, which is a database of biological networks extracted from the literature using text mining techniques.

License file LICENSE

Depends R (>= 4.4.0), MSstats

Imports httr, jsonlite, r2r, tidyr, htmlwidgets, grDevices, stats, text2vec, stopwords, xml2, rentrez

Suggests data.table, BiocStyle, knitr, rmarkdown, testthat (>= 3.0.0), mockery, MSstatsConvert, shiny

VignetteBuilder knitr

biocViews ImmunoOncology, MassSpectrometry, Proteomics, Software, QualityControl, NetworkEnrichment, Network

Encoding UTF-8

URL http://msstats.org, https://vitek-lab.github.io/MSstatsBioNet/

BugReports https://groups.google.com/forum/#!forum/msstats

Config/testthat/edition 3

RoxygenNote 7.3.3

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

git_branch devel

git_last_commit a26e6fb

git_last_commit_date 2026-03-31

Repository Bioconductor 3.23

Date/Publication 2026-04-05

Author Anthony Wu [aut, cre] (ORCID: <https://orcid.org/0009-0001-7391-9902>),
Olga Vitek [aut] (ORCID: <https://orcid.org/0000-0003-1728-1104>)

Maintainer Anthony Wu <wu.anthon@northeastern.edu>

Contents

`.populateHgncIdsInDataFrame`
Populate HGNC IDs in Data Frame

Description

This function populates the HGNC IDs in the data frame based on the Uniprot IDs.

Usage

```
.populateHgncIdsInDataFrame(df, proteinIdType)
```

Arguments

`df`	A data frame containing protein information.
`proteinIdType`	A character string specifying the type of protein ID. It can be either "Uniprot", "Uniprot_Mnemonic", or "Hgnc_Name".

Value

A data frame with populated HGNC IDs.

.populateHgncNamesInDataFrame *Populate HGNC Names in Data Frame*

Description

This function populates the HGNC names in the data frame based on the HGNC IDs.

Usage

```
.populateHgncNamesInDataFrame(df)
```

Arguments

df	A data frame containing protein information.

Value

A data frame with populated HGNC names.

.populateKinaseInfoInDataFrame *Populate Kinase Info in Data Frame*

Description

This function populates the kinase information in the data frame based on the HGNC names.

Usage

```
.populateKinaseInfoInDataFrame(df)
```

Arguments

df	A data frame containing protein information.

Value

A data frame with populated kinase information.

.populatePhophataseInfoInDataFrame *Populate Phosphatase Info in Data Frame*

Description

This function populates the phosphatase information in the data frame based on the HGNC names.

Usage

```
.populatePhophataseInfoInDataFrame(df)
```

Arguments

df	A data frame containing protein information.

Value

A data frame with populated phosphatase information.

.populateTranscriptionFactorInfoInDataFrame *Populate Transcription Factor Info in Data Frame*

Description

This function populates the transcription factor information in the data frame based on the HGNC names.

Usage

```
.populateTranscriptionFactorInfoInDataFrame(df)
```

Arguments

df	A data frame containing protein information.

Value

A data frame with populated transcription factor information.

.populateUniprotIdsInDataFrame

Populate Uniprot IDs in Data Frame

Description

This function populates the Uniprot IDs in the data frame based on the protein ID type.

Usage

```
.populateUniprotIdsInDataFrame(df, proteinIdType)
```

Arguments

df	A data frame containing protein information.
proteinIdType	A character string specifying the type of protein ID. It can be either "Uniprot" or "Uniprot_Mnemonic".

Value

A data frame with populated Uniprot IDs.

.validateAnnotateProteinInfoFromIndraInput

Validate Annotate Protein Info Input

Description

This function validates the input data frame for the annotateProteinInfoFromIndra function.

Usage

```
.validateAnnotateProteinInfoFromIndraInput(df)
```

Arguments

df	A data frame containing protein information.

Value

None. Throws an error if validation fails.

`annotateProteinInfoFromIndra` *Annotate Protein Information from Indra*

Description

This function annotates a data frame with protein information from Indra.

Usage

```
annotateProteinInfoFromIndra(df, proteinIdType)
```

Arguments

`df`	output of `groupComparison` function's comparisonResult table, which contains a list of proteins and their corresponding p-values, logFCs, along with additional HGNC ID and HGNC name columns
`proteinIdType`	A character string specifying the type of protein ID. It can be either "Uniprot", "Uniprot_Mnemonic", or "Hgnc_Name".

Value

A data frame with the following columns:

Protein Character. The original protein identifier.

UniprotID Character. The Uniprot ID of the protein.

HgncID Character. The HGNC ID of the protein.

HgncName Character. The HGNC name of the protein.

IsTranscriptionFactor Logical. Indicates if the protein is a transcription factor.

IsKinase Logical. Indicates if the protein is a kinase.

IsPhosphatase Logical. Indicates if the protein is a phosphatase.

Examples

```
df <- data.frame(Protein = c("CLH1_HUMAN"))
annotated_df <- annotateProteinInfoFromIndra(df, "Uniprot_Mnemonic")
head(annotated_df)
```

`cytoscapeNetwork` *Render a Cytoscape network visualisation*

Description

Creates an interactive network diagram powered by Cytoscape.js and the dagre layout algorithm. Nodes can carry log fold-change (logFC) values which are mapped to a blue-grey-red colour gradient. PTM (post-translational modification) site information is shown as small satellite nodes and edge overlaps are surfaced as hover tooltips.

Usage

```
cytoscapeNetwork(
  nodes,
  edges = data.frame(),
  displayLabelType = "id",
  nodeFontSize = 12,
  layoutOptions = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`nodes`	Data frame with at minimum an `id` column. Optional columns: `logFC` (numeric), `hgncName` (character), `Site` (character, underscore-separated PTM site list).
`edges`	Data frame with columns `source`, `target`, `interaction`. Optional: `site`, `evidenceLink`.
`displayLabelType`	`"id"` (default) or `"hgncName"` – controls which column is used as the visible node label.
`nodeFontSize`	Font size (px) for node labels. Default `12`.
`layoutOptions`	Named list of dagre layout options to override the defaults (e.g. `list(rankDir = "LR")`).
`width, height`	Widget dimensions passed to `createWidget`.
`elementId`	Optional explicit HTML element id.

Value

An `htmlwidget` object that renders in R Markdown, Shiny, or the RStudio Viewer pane.

Examples

```
## Not run:
nodes <- data.frame(
  id    = c("TP53", "MDM2", "CDKN1A"),
  logFC = c(1.5, -0.8, 2.1),
  stringsAsFactors = FALSE
)
edges <- data.frame(
  source      = c("TP53",  "MDM2"),
  target      = c("MDM2",  "TP53"),
  interaction = c("Activation", "Inhibition"),
  stringsAsFactors = FALSE
)
cytoscapeNetwork(nodes, edges)

## End(Not run)
```

---

cytoscapeNetworkOutput

Shiny output binding for cytoscapeNetwork

Description

Creates a Shiny output binding for a Cytoscape network visualization, allowing the network to be rendered within Shiny applications.

Usage

```
cytoscapeNetworkOutput(outputId, width = "100%", height = "500px")
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like "100%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.

Value

A Shiny output binding for a Cytoscape network visualization.

Examples

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

ui <- fluidPage(
  cytoscapeNetworkOutput("cytoNetwork")
```

```
)

server <- function(input, output, session) {
  output$cytoNetwork <- renderCytoscapeNetwork({
    nodes <- data.frame(
      id = c("TP53", "MDM2", "CDKN1A"),
      logFC = c(1.5, -0.8, 2.1),
      stringsAsFactors = FALSE
    )
    edges <- data.frame(
      source = c("TP53", "MDM2"),
      target = c("MDM2", "TP53"),
      interaction = c("Activation", "Inhibition"),
      stringsAsFactors = FALSE
    )
    cytoscapeNetwork(nodes, edges)
  })
}

shinyApp(ui, server)

## End(Not run)
```

`exportNetworkToHTML` *Export network data with Cytoscape visualization*

Description

Convenience function that takes nodes and edges data directly and creates both the configuration and HTML export in one step.

Usage

```
exportNetworkToHTML(
  nodes,
  edges,
  filename = "network_visualization.html",
  displayLabelType = "id",
  nodeFontSize = 12,
  ...
)
```

Arguments

`nodes`	Data frame with at minimum an `id` column. Optional columns: `logFC` (numeric), `hgncName` (character), `Site` (character, underscore-separated PTM site list).

edges	Data frame with columns source, target, interaction. Optional: site, evidenceLink.
filename	Output HTML filename
displayLabelType	"id" (default) or "hgncName" – controls which column is used as the visible node label.
nodeFontSize	Font size (px) for node labels. Default 12.
...	Additional arguments passed to exportCytoscapeToHTML()

Value

Invisibly returns the file path of the created HTML file

`filterSubnetworkByContext` *Filter a subnetwork by contextual relevance*

Description

Fetches PubMed abstracts for evidence PMIDs, scores each abstract against a user-supplied query, and returns only the nodes, edges, and evidence rows whose abstracts meet the scoring cutoff.

Usage

```
filterSubnetworkByContext(
  nodes,
  edges,
  query,
  cutoff = NULL,
  method = c("tag_count", "cosine")
)
```

Arguments

nodes	A dataframe of network nodes.
edges	A dataframe of network edges with columns: source, target, interaction, site, evidenceLink, stmt_hash.
query	For method = "tag_count": a character vector of tags, e.g. c("CHEK1", "DNA damage", "DNA damage repair"). For method = "cosine": a single character string.
cutoff	Numeric threshold applied to the chosen scoring method. • "tag_count": integer >= 0; abstracts must contain at least this many tags. Max possible value is length(query). Default 1. • "cosine": numeric in [-1, 1]; abstracts must score >= this value. Default 0.10.
method	One of "tag_count" (default) or "cosine".

Details

Two scoring methods are available, controlled by the `method` argument:

`"tag_count"` **(default)** Counts how many tags from `query` appear as substrings in the abstract (case-insensitive). The score for each abstract is an integer in `[0, length(query)]`. Set `cutoff` to the minimum number of tags that must appear - e.g. `cutoff = 2` keeps abstracts that mention at least 2 of your tags. `query` must be a character *vector* of tags when using this method.

`"cosine"` Scores abstracts using TF-IDF cosine similarity against `query`. Scores are in `[-1, 1]` (in practice `[0, 1]` for text). Set `cutoff` to a decimal threshold - e.g. `cutoff = 0.10`. `query` should be a single character string; expand it with synonyms and related terms for better recall under exact token matching.

Value

A named list with three elements:

`nodes`	Filtered nodes dataframe (only nodes present in kept edges)
`edges`	Filtered edges dataframe
`evidence`	Dataframe with columns: source, target, interaction, site, evidenceLink, stmt_hash, text, pmid, score. The `score` column contains tag counts (integer) or cosine similarities (numeric) depending on the method used.

`getSubnetworkFromIndra`

Get subnetwork from INDRA database

Description

Using differential abundance results from MSstats, this function retrieves a subnetwork of protein interactions from INDRA database.

Usage

```
getSubnetworkFromIndra(
  input,
  protein_level_data = NULL,
  pvalueCutoff = NULL,
  statement_types = NULL,
  paper_count_cutoff = 1,
  evidence_count_cutoff = 1,
  correlation_cutoff = 0.3,
  sources_filter = NULL,
  logfc_cutoff = NULL,
  force_include_other = NULL,
  filter_by_curation = FALSE,
```

```
  filter_by_ptm_site = FALSE,
  include_infinite_fc = FALSE,
  direction = c("both", "up", "down")
)
```

Arguments

input	output of groupComparison function's comparisionResult table, which contains a list of proteins and their corresponding p-values, logFCs, along with additional HGNC ID and HGNC name columns
protein_level_data	output of the dataProcess function's ProteinLevelData table, which contains a list of proteins and their corresponding abundances. Used for annotating correlation information and applying correlation cutoffs.
pvalueCutoff	p-value cutoff for filtering. Default is NULL, i.e. no filtering
statement_types	list of interaction types to filter on. Equivalent to statement type in INDRA. Default is NULL.
paper_count_cutoff	number of papers to filter on. Default is 1.
evidence_count_cutoff	number of evidence to filter on for each paper. E.g. A paper may have 5 sentences describing the same interaction vs 1 sentence. Default is 1.
correlation_cutoff	if protein_level_abundance is not NULL, apply a cutoff for edges with correlation less than a specified cutoff. Default is 0.3
sources_filter	filtering only on specific sources. Default is no filter, i.e. NULL. Otherwise, should be a list, e.g. c('reach', 'medscan').
logfc_cutoff	absolute log fold change cutoff for filtering proteins. Only proteins with |logFC| greater than this value will be retained. Default is NULL, i.e. no logFC filtering.
force_include_other	character vector of identifiers to include in the network, regardless if those ids are in the input data. Should be formatted as "namespace:identifier", e.g. "HGNC:1234" or "CHEBI:4911".
filter_by_curation	logical, whether to filter out statements that have been curated as incorrect in INDRA. Default is FALSE.
filter_by_ptm_site	logical, whether to filter edges based on whether the site information from INDRA matches with the PTM site in the input. Default is FALSE. Only applicable for differential PTM abundance results.
include_infinite_fc	logical, whether to include proteins with infinite log fold change (i.e. proteins that are only detected in one condition). Default is FALSE.
direction	Character string specifying the direction of regulation to include. One of "both" (default), "up" (upregulated only), or "down" (downregulated only).

Value

list of 2 data.frames, nodes and edges

Examples

```
input <- data.table::fread(system.file(
    "extdata/groupComparisonModel.csv",
    package = "MSstatsBioNet"
))
subnetwork <- getSubnetworkFromIndra(input)
head(subnetwork$nodes)
head(subnetwork$edges)
```

---

previewNetworkInBrowser

Preview network in browser

Description

Generates a temporary HTML file for the network visualization and opens it in the default web browser for quick preview.

Usage

```
previewNetworkInBrowser(
  nodes,
  edges,
  displayLabelType = "id",
  nodeFontSize = 12
)
```

Arguments

nodes	Data frame with at minimum an `id` column. Optional columns: `logFC` (numeric), `hgncName` (character), `Site` (character, underscore-separated PTM site list).
edges	Data frame with columns `source`, `target`, `interaction`. Optional: `site`, `evidenceLink`.
displayLabelType	`"id"` (default) or `"hgncName"` – controls which column is used as the visible node label.
nodeFontSize	Font size (px) for node labels. Default `12`.

Value

Invisibly returns the file path of the temporary HTML file.

Examples

```
## Not run:
nodes <- data.frame(id = c("A", "B", "C"))
edges <- data.frame(source = c("A", "B"), target = c("B", "C"))
previewNetworkInBrowser(nodes, edges)

## End(Not run)
```

`renderCytoscapeNetwork` *Render a Cytoscape network in a Shiny application. This function is used to render a Cytoscape network visualization within a Shiny application.*

Description

Render a Cytoscape network in a Shiny application. This function is used to render a Cytoscape network visualization within a Shiny application.

Usage

```
renderCytoscapeNetwork(expr, env = parent.frame())
```

Arguments

`expr`	An expression that generates an HTML widget (or a promise of an HTML widget).
`env`	The environment in which to evaluate `expr`.

Value

A rendered Cytoscape network widget for use in Shiny applications.

Examples

```
## Not run:
library(shiny)
library(MSstatsBioNet)

ui <- fluidPage(
  cytoscapeNetworkOutput("cytoNetwork")
)

server <- function(input, output, session) {
  output$cytoNetwork <- renderCytoscapeNetwork({
    nodes <- data.frame(
      id    = c("TP53", "MDM2", "CDKN1A"),
```

```
      logFC = c(1.5, -0.8, 2.1),
      stringsAsFactors = FALSE
    )
    edges <- data.frame(
      source      = c("TP53",  "MDM2"),
      target      = c("MDM2",  "TP53"),
      interaction = c("Activation", "Inhibition"),
      stringsAsFactors = FALSE
    )
    cytoscapeNetwork(nodes, edges)
  })
}

shinyApp(ui, server)

## End(Not run)
```

Index