

# Package ‘vissE’

April 24, 2025

**Title** Visualising Set Enrichment Analysis Results

**Version** 1.16.0

**Description** This package enables the interpretation and analysis of results from a gene set enrichment analysis using network-based and text-mining approaches. Most enrichment analyses result in large lists of significant gene sets that are difficult to interpret. Tools in this package help build a similarity-based network of significant gene sets from a gene set enrichment analysis that can then be investigated for their biological function using text-mining approaches.

**biocViews** Software, GeneExpression, GeneSetEnrichment,  
NetworkEnrichment, Network

**License** GPL-3

**Encoding** UTF-8

**LazyDataCompression** bzip2

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Depends** R (>= 4.1)

**Imports** igraph, methods, plyr, ggplot2, scico, RColorBrewer, tm,  
ggwordcloud, GSEABase, reshape2, grDevices, ggforce, msigdb,  
ggrepel, textstem, tidygraph, stats, scales, ggraph

**Suggests** testthat, org.Hs.eg.db, org.Mm.eg.db, patchwork, singscore,  
knitr, rmarkdown, prettydoc, BiocStyle, pkgdown, covr

**URL** <https://davislaboratory.github.io/vissE>

**BugReports** <https://github.com/DavisLaboratory/vissE/issues>

**VignetteBuilder** knitr

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

**git\_branch** RELEASE\_3\_21

**git\_last\_commit** a90700e

**git\_last\_commit\_date** 2025-04-15

**Repository** Bioconductor 3.21

**Date/Publication** 2025-04-23

**Author** Dharmesh D. Bhuva [aut, cre] (ORCID:  
<https://orcid.org/0000-0002-6398-9157>),  
 Ahmed Mohamed [ctb]

**Maintainer** Dharmesh D. Bhuva <bhuva.d@wehi.edu.au>

## Contents

vissE-package . . . . .	2
bhuvad_theme . . . . .	3
characteriseGeneset . . . . .	4
computeMsigNetwork . . . . .	5
computeMsigOverlap . . . . .	6
computeMsigWordFreq . . . . .	7
findMsigClusters . . . . .	8
getMsigExclusionList . . . . .	9
hgsc . . . . .	10
plotGeneStats . . . . .	10
plotMsigNetwork . . . . .	12
plotMsigPPI . . . . .	13
plotMsigWordcloud . . . . .	15

<b>Index</b>	<b>17</b>
--------------	-----------

---

vissE-package	<i>vissE: Visualising Set Enrichment Analysis Results</i>
---------------	---

---

## Description

This package enables the interpretation and analysis of results from a gene set enrichment analysis using network-based and text-mining approaches. Most enrichment analyses result in large lists of significant gene sets that are difficult to interpret. Tools in this package help build a similarity-based network of significant gene sets from a gene set enrichment analysis that can then be investigated for their biological function using text-mining approaches.

## Details

This package supports four workflows to enhance gene set enrichment analysis:

1. Clustering results from a gene set enrichment analysis (e.g. using `limma::fry`, `singscore` or GSEA). The functions required for this analysis are `computeMsigOverlap`, `computeMsigNetwork` and `plotMsigNetwork`.
2. Interpreting gene set clusters (identified in the first analysis) by performing text-mining of gene set names and descriptions. The main function required to perform text-mining of gene sets is `plotMsigWordcloud`. Other functions can be used to access intermediate results.
3. Visualise gene-level statistics for gene set clusters identified in the first analysis to link back gene set clusters to the genes of interest. This can be done using the `plotGeneStats` function.
4. Identifying gene sets similar to a list of genes identified from a DE analysis using set overlap measures. This can be done using the `characteriseGeneset` function.

**Author(s)**

**Maintainer:** Dharmesh D. Bhuva <bhuva.d@wehi.edu.au> ([ORCID](#))

Other contributors:

- Ahmed Mohamed <mohamed.a@wehi.edu.au> [contributor]

**See Also**

Useful links:

- <https://davislaboratory.github.io/vissE>
- Report bugs at <https://github.com/DavisLaboratory/vissE/issues>

---

bhuvad\_theme

*Custom theme*

---

**Description**

Custom theme

**Usage**

```
bhuvad_theme(r1 = 1.1)
```

**Arguments**

`r1` a numeric, scaling factor to apply to text sizes

**Value**

a ggplot2 theme

**Examples**

```
p1 = ggplot2::ggplot()
p1 + bhuvad_theme()
```

---

characteriseGeneset    *Functionally characterise a list of genes*

---

### Description

This function can be used to perform a network-based enrichment analysis of a list of genes. The list of genes are characterised based on their similarity with gene sets from the MSigDB. A network of similar gene sets is retrieved using this function.

### Usage

```
characteriseGeneset(  
  gs,  
  thresh = 0.2,  
  measure = c("overlapcoef", "jaccard"),  
  gscolcs = c("h", "c2", "c5"),  
  org = c("auto", "hs", "mm")  
)
```

### Arguments

gs	a GeneSet object, representing the list of genes that need to be characterised.
thresh	a numeric, specifying the threshold to discard pairs of gene sets.
measure	a character, specifying the similarity measure to use: <code>ari</code> for the Adjusted Rand Index, <code>jaccard</code> for the Jaccard Index and <code>overlapcoef</code> for the Overlap Coefficient.
gscolcs	a character, listing the MSigDB collections to use as a background (defaults to <code>h</code> , <code>c2</code> , and <code>c5</code> ). Collection types can be retrieved using <code>msigdb::listCollections()</code> .
org	a character, specifying the organism to use. This can either be "auto" (default), "hs" or "mm".

### Value

an `igraph` object, containing gene sets that are similar to the query set. The network contains relationships between results of the query too.

### Examples

```
library(GSEABase)  
data(hgsc)  
  
#create a geneset using one of the Hallmark gene sets  
mySet <- GeneSet(  
  geneIds(hgsc[[2]]),  
  setName = 'MySet',  
  geneIdType = SymbolIdentifier()  
)
```

```
#characterise the custom gene set
ig <- characteriseGeneset(mySet)
plotMsigNetwork(ig)
```

---

computeMsigNetwork      *Compute a network using computed gene set overlap*

---

### Description

Computes an igraph object using information on gene sets and gene sets computed using the [computeMsigOverlap\(\)](#) function.

### Usage

```
computeMsigNetwork(genesetOverlap, msigGsc)
```

### Arguments

`genesetOverlap` a data.frame, containing results of an overlap analysis computed using the [computeMsigOverlap\(\)](#) function.

`msigGsc` a GeneSetCollection object, containing gene sets used to compute overlap.

### Value

an igraph object

### Examples

```
data(hgsc)
ovlap <- computeMsigOverlap(hgsc)
ig <- computeMsigNetwork(ovlap, hgsc)
```

---

computeMsigOverlap      *Compute gene set overlap*

---

### Description

Compute overlap between gene sets from a GeneSetCollection using the Jaccard index or the overlap coefficient. These values can then be used to compute a network of gene set overlaps.

### Usage

```
computeMsigOverlap(  
  msigGsc1,  
  msigGsc2 = NULL,  
  thresh = 0.25,  
  measure = c("ari", "jaccard", "overlapcoef")  
)
```

### Arguments

msigGsc1	a GeneSetCollection object.
msigGsc2	a GeneSetCollection object or NULL if pairwise overlaps are to be computed.
thresh	a numeric, specifying the threshold to discard pairs of gene sets.
measure	a character, specifying the similarity measure to use: <code>ari</code> for the Adjusted Rand Index, <code>jaccard</code> for the Jaccard Index and <code>overlapcoef</code> for the Overlap Coefficient.

### Value

a data.frame, containing the overlap structure of gene sets represented as a network in the simple interaction format (SIF).

### Examples

```
data(hgsc)  
ovlap <- computeMsigOverlap(hgsc)
```

---

computeMsigWordFreq    *Compute word frequencies for a single MSigDB collection*

---

## Description

Compute word frequencies for a single MSigDB collection

## Usage

```
computeMsigWordFreq(
  msigGsc,
  weight = NULL,
  measure = c("tfidf", "tf"),
  version = msigdb::getMsigdbVersions(),
  org = c("auto", "hs", "mm"),
  rmwords = getMsigExclusionList(),
  idf = NULL
)
```

## Arguments

msigGsc	a GeneSetCollection object, containing gene sets from the MSigDB. The <a href="#">GSEABase::getBroadSets()</a> function can be used to parse XML files downloaded from MSigDB.
weight	a named numeric vector, containing weights to apply to each gene-set. This can be $-\log_{10}(\text{FDR})$ , $-\log_{10}(\text{p-value})$ or an enrichment score (ideally unsigned).
measure	a character, specifying how frequencies should be computed. "tf" uses term frequencies and "tfidf" (default) applies inverse document frequency weights to term frequencies.
version	a character, specifying the version of msigdb to use (see <a href="#">msigdb::getMsigdbVersions()</a> ).
org	a character, specifying the organism to use. This can either be "auto" (default), "hs" or "mm".
rmwords	a character vector, containing an exclusion list of words to discard from the analysis.
idf	a list of named numeric vectors, specifying inverse document frequencies to use to penalise terms from gene-set names and short descriptions. This should be a vector of length 2 with names "Name" and "Short". Numeric vectors should contain weights and names should represent the term. Precomputed versions can be retrieved using the <a href="#">msigdb::getMsigdbIDF()</a> .

## Value

a list, containing two data.frames summarising the results of the frequency analysis on gene set names and short descriptions.

**Examples**

```
data(hgsc)
freq <- computeMsigWordFreq(hgsc, measure = 'tfidf')
```

---

findMsigClusters	<i>Identify gene-set clusters from a gene-set overlap network</i>
------------------	---

---

**Description**

This function identifies gene-set clusters from a gene-set overlap network produced using `vissE`. Various graph clustering algorithms from the `igraph` package can be used for clustering. Gene-set clusters identified are then sorted based on their size and a given statistic of interest (absolute of the statistic is maximised per cluster).

**Usage**

```
findMsigClusters(
  ig,
  genesetStat = NULL,
  minSize = 2,
  alg = igraph::cluster_walktrap,
  alparams = list()
)
```

**Arguments**

<code>ig</code>	an <code>igraph</code> object, containing a network of gene set overlaps computed using <code>computeMsigNetwork()</code> .
<code>genesetStat</code>	a named numeric, containing statistics for each gene-set that are to be used in cluster prioritisation. If <code>NULL</code> , clusters are prioritised based on their size (number of gene-sets in them).
<code>minSize</code>	a numeric, stating the minimum size a cluster can be (default is 2).
<code>alg</code>	a function, from the <code>igraph</code> package that should be used to perform graph-clustering (default is <code>igraph::cluster_walktrap</code> ). The function should produce a <code>communities</code> object.
<code>alparams</code>	a list, specifying additional parameters that are to be passed to the graph clustering algorithm.

**Details**

Gene-sets clusters are identified using graph clustering and are prioritised based on a combination of cluster size and optionally, a statistic of interest (e.g., enrichment scores). A product-of-ranks approach is used to prioritise clusters when gene-set statistics are available. In this approach, clusters are ranked based on their cluster size (largest to smallest) and on the median absolute statistic



of gene-sets within it (largest to smallest). The product of these ranks is computed and clusters are ranked based on these product-of-rank statistic (smallest to largest).

When prioritising using cluster size and gene-set statistics, if statistics for some gene-sets in the network are missing, only the size is used in cluster prioritisation.

**Value**

a list, containing gene-sets that belong to each cluster. Items in the list are organised based on prioritisation.

**Examples**

```
data(hgsc)
overlap <- computeMsigOverlap(hgsc, thresh = 0.25)
ig <- computeMsigNetwork(overlap, hgsc)
findMsigClusters(ig)
```

---

getMsigExclusionList *Exclusion list of words for MSigDB gene set text mining*

---

**Description**

List of words to discard when performing text mining MSigDB gene set names and short descriptions.

**Usage**

```
getMsigExclusionList(custom = c())
```

**Arguments**

custom            a character vector, containing list of words to add onto existing exclusion list.

**Value**

a character vector, containing words to be excluded from the text mining analysis.

**Examples**

```
getMsigExclusionList('remove')
```

---

hgsc

*The Hallmark collection from the MSigDB*

---

### Description

The molecular signatures database (MSigDB) is a collection of over 25000 gene expression signatures. Signatures in v7.2 are divided into 9 categories. The Hallmarks collection contains gene expression signatures representing molecular processes that are hallmarks in cancer development and progression.

### Usage

hgsc

### Format

A GeneSetCollection object with 50 GeneSet objects representing the 50 Hallmark gene expression signatures.

### References

Subramanian, A., Tamayo, P., Mootha, V. K., Mukherjee, S., Ebert, B. L., Gillette, M. A., ... & Mesirov, J. P. (2005). Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences*, 102(43), 15545-15550.

Liberzon, A., Subramanian, A., Pinchback, R., Thorvaldsdóttir, H., Tamayo, P., & Mesirov, J. P. (2011). Molecular signatures database (MSigDB) 3.0. *Bioinformatics*, 27(12), 1739-1740.

Liberzon, A., Birger, C., Thorvaldsdóttir, H., Ghandi, M., Mesirov, J. P., & Tamayo, P. (2015). The molecular signatures database hallmark gene set collection. *Cell systems*, 1(6), 417-425.

---

plotGeneStats

*Plot gene statistics for clusters of gene sets*

---

### Description

This function plots gene statistics against gene frequencies for any given cluster of gene sets. The plot can be used to identify genes that are over-represented in a cluster of gene-sets (identified based on gene-set overlaps) and have a strong statistic (e.g. log fold-change or p-value).

**Usage**

```
plotGeneStats(  
  geneStat,  
  msigGsc,  
  groups,  
  statName = "Gene-level statistic",  
  topN = 5  
)
```

**Arguments**

geneStat	a named numeric, containing the statistic to be displayed. The vector must be named with either gene Symbols or Entrez IDs depending on annotations in msigGsc.
msigGsc	a GeneSetCollection object, containing gene sets from the MSigDB. The <a href="#">GSEABase::getBroadSets()</a> function can be used to parse XML files downloaded from MSigDB.
groups	a named list, of character vectors or numeric indices specifying node groupings. Each element of the list represent a group and contains a character vector with node names.
statName	a character, specifying the name of the statistic.
topN	a numeric, specifying the number of genes to label. The top genes are those with the largest count and statistic.

**Value**

a ggplot object, plotting the gene-level statistic against gene frequencies in the cluster of gene sets.

**Examples**

```
library(GSEABase)  
  
data(hgsc)  
groups <- list('g1' = names(hgsc)[1:25], 'g2' = names(hgsc)[26:50])  
  
#create statistics  
allgenes = unique(unlist(geneIds(hgsc)))  
gstats = rnorm(length(allgenes))  
names(gstats) = allgenes  
  
#plot  
plotGeneStats(gstats, hgsc, groups)
```

---

plotMsigNetwork      *Plot a gene set overlap network*

---

### Description

Plots a network of gene set overlap with overlap computed using the [computeMsigOverlap\(\)](#) and a graph created using [computeMsigNetwork\(\)](#).

### Usage

```
plotMsigNetwork(
  ig,
  markGroups = NULL,
  genesetStat = NULL,
  nodeSF = 1,
  edgeSF = 1,
  lytFunc = "graphopt",
  lytParams = list(),
  rmUnmarkedGroups = FALSE,
  maxGrp = 12
)
```

### Arguments

ig	an igraph object, containing a network of gene set overlaps computed using <a href="#">computeMsigNetwork()</a> .
markGroups	a named list, of character vectors. Each element of the list represent a group and contains a character vector with node names. Up to 12 groups can be visualised in the plot.
genesetStat	a named numeric, statistic to project onto the nodes. These could be p-values, log fold-changes or gene set score from a singscore-based analysis.
nodeSF	a numeric, indicating the scaling factor to apply to node sizes.
edgeSF	a numeric, indicating the scaling factor to apply to edge widths.
lytFunc	a character, specifying the layout to use (see <a href="#">ggraph::create_layout()</a> ).
lytParams	a named list, containing additional parameters needed for the layout (see <a href="#">ggraph::create_layout()</a> ).
rmUnmarkedGroups	a logical, indicating whether unmarked groups should be removed from the network (TRUE) or retained (FALSE - default).
maxGrp	a numeric, specifying the maximum number of groups to plot.

### Value

a ggplot2 object

## Examples

```
data(hgsc)
ovlap <- computeMsigOverlap(hgsc, thresh = 0.15)
ig <- computeMsigNetwork(ovlap, hgsc)
groups <- list(
  'g1' = c("HALLMARK_HYPOXIA", "HALLMARK_GLYCOLYSIS"),
  'g2' = c("HALLMARK_INTERFERON_GAMMA_RESPONSE")
)

plotMsigNetwork(ig, markGroups = groups)
```

---

plotMsigPPI

*Plot PPI network for gene-set clusters identified using vissE*

---

## Description

This function plots the protein-protein interaction (PPI) network for a gene-set cluster identified using *vissE*. The international molecular exchange (IMEx) PPI is used to obtain PPIs for genes present in a gene-set cluster.

## Usage

```
plotMsigPPI(
  ppidf,
  msigGsc,
  groups,
  geneStat = NULL,
  statName = "Gene-level statistic",
  threshConfidence = 0,
  threshFrequency = 0.25,
  threshStatistic = 0,
  threshUseAbsolute = TRUE,
  topN = 5,
  nodeSF = 1,
  edgeSF = 1,
  lytFunc = "graphopt",
  lytParams = list()
)
```

## Arguments

**ppidf** a data.frame, containing a protein-protein interaction from the IMEx database. This can be retrieved from the `msigdb::getIMEX()` function.

**msigGsc** a GeneSetCollection object, containing gene sets from the MSigDB. The `GSEABase::getBroadSets()` function can be used to parse XML files downloaded from MSigDB.

<code>groups</code>	a named list, of character vectors or numeric indices specifying node groupings. Each element of the list represent a group and contains a character vector with node names.
<code>geneStat</code>	a named numeric, containing the statistic to be displayed. The vector must be named with either gene Symbols or Entrez IDs depending on annotations in <code>msigGsc</code> .
<code>statName</code>	a character, specifying the name of the statistic.
<code>threshConfidence</code>	a numeric, specifying the confidence threshold to apply to determine high confidence interactions. This should be a value between 0 and 1 (default is 0).
<code>threshFrequency</code>	a numeric, specifying the frequency threshold to apply to determine more frequent genes in the gene-set cluster. The frequency of a gene is computed as the proportion of gene-sets to which the gene belongs. This should be a value between 0 and 1 (default is 0.25).
<code>threshStatistic</code>	a numeric, specifying the threshold to apply to gene-level statistics (e.g. a log fold-change). This should be a value between 0 and 1 (default is 0).
<code>threshUseAbsolute</code>	a logical, indicating whether the <code>threshStatistic</code> threshold should be applied to absolute values (default TRUE). This can be used to threshold on statistics such as the log fold-change from a differential expression analysis.
<code>topN</code>	a numeric, specifying the number of genes to label. The top genes are those with the largest count and statistic.
<code>nodeSF</code>	a numeric, indicating the scaling factor to apply to node sizes.
<code>edgeSF</code>	a numeric, indicating the scaling factor to apply to edge widths.
<code>lytFunc</code>	a character, specifying the layout to use (see <code>ggraph::create_layout()</code> ).
<code>lytParams</code>	a named list, containing additional parameters needed for the layout (see <code>ggraph::create_layout()</code> ).

**Value**

a `ggplot` object with the protein-protein interaction networks plot for each gene-set cluster.

**Examples**

```
data(hgsc)
grps = list('early' = 'HALLMARK_ESTROGEN_RESPONSE_EARLY', 'late' = 'HALLMARK_ESTROGEN_RESPONSE_LATE')
ppi = msigdb::getIMEX(org = 'hs', inferred = TRUE)
plotMsigPPI(ppi, hgsc, grps)
```

---

plotMsigWordcloud      *Compute and plot word frequencies for multiple MSigDB collections*

---

## Description

Given a gene set collection, this function computes the word frequency of gene set names from the Molecular Signatures Database (MSigDB) collection (split by `_`). Word frequencies are also computed using short descriptions attached with each gene set object.

## Usage

```
plotMsigWordcloud(  
  msigGsc,  
  groups,  
  weight = NULL,  
  measure = c("tfidf", "tf"),  
  version = msigdb::getMsigdbVersions(),  
  org = c("auto", "hs", "mm"),  
  rmwords = getMsigExclusionList(),  
  type = c("Name", "Short"),  
  idf = NULL  
)
```

## Arguments

msigGsc	a GeneSetCollection object, containing gene sets from the MSigDB. The <code>GSEABase::getBroadSets()</code> function can be used to parse XML files downloaded from MSigDB.
groups	a named list, of character vectors or numeric indices specifying node groupings. Each element of the list represent a group and contains a character vector with node names.
weight	a named numeric vector, containing weights to apply to each gene-set. This can be $-\log_{10}(\text{FDR})$ , $-\log_{10}(\text{p-value})$ or an enrichment score (ideally unsigned).
measure	a character, specifying how frequencies should be computed. "tf" uses term frequencies and "tfidf" (default) applies inverse document frequency weights to term frequencies.
version	a character, specifying the version of msigdb to use (see <code>msigdb::getMsigdbVersions()</code> ).
org	a character, specifying the organism to use. This can either be "auto" (default), "hs" or "mm".
rmwords	a character vector, containing an exclusion list of words to discard from the analysis.
type	a character, specifying the source of text mining. Either gene set names (Name) or descriptions (Short) can be used.

`idf` a list of named numeric vectors, specifying inverse document frequencies to use to penalise terms from gene-set names and short descriptions. This should be a vector of length 2 with names "Name" and "Short". Numeric vectors should contain weights and names should represent the term. Precomputed versions can be retrieved using the `msigdb::getMsigdbIDF()`.

**Value**

a ggplot object.

**Examples**

```
data("hgsc")
groups <- list('g1' = names(hgsc)[1:25], 'g2' = names(hgsc)[26:50])
plotMsigWordcloud(hgsc, groups, rmwords = getMsigExclusionList())
```



# Index

## \* datasets

hgsc, [10](#)

## \* internal

vissE-package, [2](#)

bhuvad\_theme, [3](#)

characteriseGeneset, [2](#), [4](#)

computeMsigNetwork, [2](#), [5](#)

computeMsigNetwork(), [8](#), [12](#)

computeMsigOverlap, [2](#), [6](#)

computeMsigOverlap(), [5](#), [12](#)

computeMsigWordFreq, [7](#)

findMsigClusters, [8](#)

getMsigExclusionList, [9](#)

GSEABase::getBroadSets(), [7](#), [11](#), [13](#), [15](#)

hgsc, [10](#)

msigdb::getIMEX(), [13](#)

msigdb::getMsigdbIDF(), [7](#), [16](#)

msigdb::listCollections(), [4](#)

plotGeneStats, [2](#), [10](#)

plotMsigNetwork, [2](#), [12](#)

plotMsigPPI, [13](#)

plotMsigWordcloud, [2](#), [15](#)

vissE (vissE-package), [2](#)

vissE-package, [2](#)