Package ‘massiveGST’

March 29, 2023

Type Package

Title Competitive Gene Sets Test with the Mann-Whitney-Wilcoxon Test

Version 1.2.3

Date 2023-03-28

Maintainer Stefano Maria Pagnotta <pagnotta@unisannio.it>

Description Friendly implementation of the Mann-Whitney-Wilcoxon test for competitive gene set enrichment analysis.

Depends R (>= 4.1.0), formattable (>= 0.2.1), msigdbr (>= 7.4.0), WriteXLS (>= 6.3.0), igraph (>= 1.2.6), visNetwork (>= 2.0.9)

Suggests knitr, rmarkdown

License GPL (>= 3)

URL <https://github.com/stefanoMP/massiveGST>, <http://www.massivegenesetstest.org/>

VignetteBuilder knitr

NeedsCompilation no

Author Stefano Maria Pagnotta [aut, cre, cph] (<https://orcid.org/0000-0002-8298-9777>)

Repository CRAN

Date/Publication 2023-03-29 16:40:02 UTC

R topics documented:

cut_by_logit2NES 2
cut_by_NES 3
cut_by_significance 4
geneSets.sim 6
get_geneProfile 7
get_geneSets_from_local_files 8
get_geneSets_from_msigdbr 9
massiveGST 10
massiveORT 11

plot.mGST 13
pre_ranked_list 15
save_as_tsv 15
save_as_xls 16
summary.mGST 17
write_geneSets_to_gmt 19

Index **20**

`cut_by_logit2NES` *Trim the table of results.*

Description

This function trims the table of results from massiveGST function retaining the rows with a logit2NES below the specified threshold.

Usage

```
cut_by_logit2NES(ttable, logit2NES_threshold = 0.58)
```

Arguments

`ttable`	a data frame of "mGST" class coming from massiveGST function.
`logit2NES_threshold`	a real value

Value

A data frame.

Note

the functions cut_by_NES, cut_by_logit2NES, and cut_by_significance can be nested.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

massiveGST, cut_by_NES, cut_by_significance,

summary.mGST, plot.mGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

head(ans)

cut_by_logit2NES(ans)
cut_by_logit2NES(cut_by_significance(ans))

plot(cut_by_logit2NES(ans))
```

`cut_by_NES` *Trim the table of results.*

Description

This function trims the table of results from massiveGST function retaining the rows with a NES below the specified threshold.

Usage

```
cut_by_NES(ttable, NES_threshold = 0.6)
```

Arguments

`ttable`	a data frame of 'mGST' class coming from massiveGST function.
`NES_threshold`	a real value between 0.0 and 1.

Value

A data frame.

Note

the functions cut_by_NES, cut_by_logit2NES, and cut_by_significance can be nested. In the case the test has alternative = 'two.sided', it is better to use cut_by_logit2NES for a symmetric trim of both directions.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

massiveGST, cut_by_logit2NES, cut_by_significance, summary.mGST, plot.mGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "greater")

head(ans)
cut_by_NES(ans, NES_threshold = .65)
summary(cut_by_NES(ans, NES_threshold = .65))
```

`cut_by_significance` *Trim the table of results.*

Description

This function trims the table of results from massiveGST function according to the significance required.

Usage

```
cut_by_significance(ttable,
  level_of_significance = 0.05,
  where = c("BH.value", "bonferroni", "p.value")
)
```

Arguments

`ttable`	a data frame of "mGST" class coming from massiveGST function.
`level_of_significance`	a real value between 0.0 and 1.
`where`	a character string specifying where the level_of_significance has to be applied to the output; must be one of "p.value", "BH.value" (default), and "bonferroni"

Details

BH.value is the adjustment of p-values according to Benijamini and Hockberg's method; B.value is the adjustment of p-values according to Bonferroni's method.

Value

A data frame.

Note

the functions cut_by_NES, cut_by_logit2NES, and cut_by_significance can be nested.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

massiveGST, cut_by_logit2NES, cut_by_NES, summary.mGST, plot.mGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

head(ans)
cut_by_significance(ans)
cut_by_significance(ans, level_of_significance = 0.05, where = "p")
cut_by_logit2NES(cut_by_significance(ans))

summary(cut_by_significance(ans, level_of_significance = 0.05, where = "bonferroni"))

plot(cut_by_significance(ans, level_of_significance = 0.05, where = "bonferroni"))
```

geneSets.sim *Compute the similarities between a collection of gene sets.*

Description

Compute the similarities between a collection of gene sets using a convex function of the Jaccard and overlap indeces.

Usage

```
geneSets.sim(gs, eps = 0.25)
```

Arguments

`gs`	a character vector of gene-sets.
`eps`	a real value between 0.0 and 1.0 controlling the contribution of the Jaccard and overlap similaties to their convex combination; eps = 0.25 (default), see details.

Details

The similarity between the gene-set is computed a convex combination of the Jaccard and overlap similarities. See the reference for further details.

Value

returns an object of class "dist", where the values are the similaties between gene sets.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

dist

Examples

```
library(massiveGST)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")[1:5]

# compute the similarities
geneSets.sim(geneSets)
ssim <- geneSets.sim(geneSets)
ssim <- as.matrix(ssim)
diag(ssim) <- 1
ssim
```

`get_geneProfile` *Load a gene-profile from a txt file.*

Description

Load a gene-profile from a txt file.

Usage

```
get_geneProfile(ffile)
```

Arguments

`ffile`	a character string or a list of a character pointing to a local file

Details

The txt file contains two columuns separated by a tabulation. The first column is the gene name (or entrez, ensembl, etc); the second column are the numeric values associated with each gene. The profile do not need to be sorted.

As an example, see the file in /massiveGST/extdata/pre_ranked_list.txt

See the path in the example below.

Value

A named list of numeric values.

Author(s)

Stefano M. Pagnotta

See Also

pre_ranked_list

Examples

```
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
fname
geneProfile <- get_geneProfile(fname)
class(geneProfile)
head(geneProfile)
tail(geneProfile)
```

---

get_geneSets_from_local_files

Load the gene-sets collection from local gmt files

Description

Load the gene-sets collection from local gmt files

Usage

```
get_geneSets_from_local_files(ffiles)
```

Arguments

`ffiles`	a character string or a list of a character pointing to local files

Value

A vector list of gene-sets

Author(s)

Stefano M. Pagnotta

See Also

get_geneSets_from_msigdbr, write_geneSets_to_gmt

Examples

```
library(massiveGST)

tmp <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

fname1 <- file.path(tempdir(), "h1.gmt")
write_geneSets_to_gmt(tmp, fileName = fname1)

fname2 <- file.path(tempdir(), "h2.gmt")
write_geneSets_to_gmt(tmp, fileName = fname2)
```

```
# getting one collection
geneSets <- get_geneSets_from_local_files(fname1)
length(geneSets)

# getting two collections
geneSets <- get_geneSets_from_local_files(c(fname1, fname2))
length(geneSets)
```

---

get_geneSets_from_msigdbr

Get the gene-sets from the msigdbr package.

Description

This is a wrapper for extraction a gene-sets collection as a vector list to match the data structure for massiveGST function.

Usage

```
get_geneSets_from_msigdbr(category, what, subcategory = NULL, species = "Homo sapiens")
```

Arguments

`category`	MSigDB collection abbreviation, such as H or C1.
`what`	a character string specifying the code representation of the genes; must be one of "gene_symbol", "entrez_gene", "ensembl_gene", "human_gene_symbol", "human_entrez_gene", "human_ensembl_gene";
`subcategory`	MSigDB sub-collection abbreviation, such as CGP or BP; NULL (default)
`species`	Species name, such as 'Homo sapiens' or 'Mus musculus'.

Value

A vector list of gene-sets

Author(s)

Stefano M. Pagnotta

See Also

msigdbr

Examples

```
library(massiveGST)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

class(geneSets)
head(geneSets, 3)
```

---

massiveGST *massive Gene-Sets Test with Mann-Whitney-Wilcoxon statistics.*

Description

Perform a competitive gene set enrichment analysis by applying the Mann-Withney-Wilcoxon test.

Usage

```
massiveGST(gene_profile, gene_sets,
  cols_to_remove = NULL,
  alternative = c("two.sided", "less", "greater")
  )
```

Arguments

gene_profile	a named list of values; the names have to match the names of genes in the gene-set.
gene_sets	a character vector of gene-sets.
cols_to_remove	a list of colnames to eventually remove from the output.
alternative	a character string specifying the alternative hypothesis of the MWW test; must be one of "two.sided" (default), "greater" or "less".

Value

A data frame with columns

size	Original size of the gene-set.
actualSize	Size of the gene-set after the match with the gene-profile.
NES	(Normalized Enrichment Score) the strength of the association of the gene-set with the gene profile; also the percentile rank of the gene-set in the universe of the genes ouside the gene-set.
odd	odd transformation of the NES.
logit2NES	logit transformation of the NES.
abs_logit2NES	absolute value of the logit2NES in the case of "two.sided" alternative.
p.value	p-values associated with the gene-set.

`BH.value`	Benijamini and Hockberg adjustment of the p.values.
`B.value`	Bonferroni adjustment of the p.values.
`relevance`	marginal ordering of the table.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

summary.mGST, plot.mGST, cut_by_logit2NES, cut_by_NES, cut_by_significance

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

ans
```

`massiveORT` *A wrapper to fisher.test to get over representation analysis of gene sets.*

Description

The function massiveORT essentially is a wrapper to the function fisher.test in charge to 1) arrange the input to feed fisher.test in sequence for each gene set, 2) arrange the output in a data frame compatible with the other function of the package, and 3) compute the universe of genes for the analysis.

Usage

```
massiveORT(gene_list, gene_sets, universe = NULL,
           alternative = c("greater", "less", "two.sided"))
```

Arguments

`gene_list`	a list of gene names, or gene ids that have to match the corresponidng in the gene-set.
`gene_sets`	a character vector of gene-sets.
`universe`	a list of gene, or gene ids, that defines the universe for the analysis (see details); NULL by default.
`alternative`	a character string specifying the alternative hypothesis of the fisher.test; must be one of "two.sided", "greater" (default) or "less".

Details

This function allows to define externally or compute the universe of reference of the analysis. By default (universe = NULL), the universe is computed starting from the gene names contributing at least once in each gene set.

Value

A data frame with columns

`universe_size`	size of the universe of genes.
`geneList_size`	size of intersection between the gene list and the universe.
`geneSet_size`	size of intersection between the gene set and the universe.
`geneList_in_GenesSet_size`	size of the intersection between the geneList and the geneSet.
`odds_ratio`	odd ratio coming from the fisher.test
`log2_odds_ratio`	log2 transformation of odds_ratio.
`p.value`	p-values associated with the gene-set coming from the fisher.test
`BH.value`	Benijamini and Hockberg adjustment of the p.values
`B.value`	Bonferroni adjustment of the p.values
`relevance`	marginal ordering of the table.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

fisher.test, cut_by_significance

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)
geneList <- names(head(geneProfile, 1000))

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "C5", subcategory = "CC", what = "gene_symbol")
geneSets <- geneSets[1:250]

# run the function
ans <- massiveORT(geneList, geneSets)
cut_by_significance(ans)

plot(cut_by_significance(ans), geneSets,as.network = TRUE)
```

`plot.mGST` *Graphical rendering of the enrichment analysis.*

Description

This function displays the enrichment analysis results both as a bar-plot and a network of gene-sets.

Usage

```
## S3 method for class 'mGST'
plot(x,
  gene_sets = NULL,
  order_by = "logit2NES",
  top = 30,
  eps = 0.25,
  as.network = FALSE,
  similarity_threshold = 1/3,
  manipulation = FALSE,
  autoResize = TRUE,
  ...
)
```

Arguments

`x`	a data structure coming from the massiveGST function
`gene_sets`	a character vector of gene-sets; mandatory for the network display
`order_by`	a character string specifying whick should be the ordering in the bar-plot; must be one of "relevance", "NES", "logit2NES" (default), "p.value", "BH.value", and "bonferroni". These are the same options of summary.mGST

`top`	an integer value controlling how many gene-sets have to be displaued in the bar-plot; top = 30 (default)
`as.network`	a logical value to switch to a network display; as.network = FALSE (default)
`similarity_threshold`	a real value to cut the similarities between gene-stes below this value; similarity_threshold = 1/3 (default)
`eps`	a real value between 0.0 and 1.0 controlling the contribution of the Jaccard and overlap similaties to their convex combination; eps = 0.25 (default), see details.
`manipulation`	a logical value allowing to manipulate the network; manipulation = FALSE (default); see visOptions
`autoResize`	a logical value allowing to resize the network; resize = TRUE (default); see visOptions
`...`	other graphical parameters

Details

This function display the results of enrichment analysis both as a bar-plot and a network.

The network rendering is with the visNetwork package.

The similarity between the gene-set is computed a convex combination of the Jaccard and overlap similarities. See the reference for further details.

Value

In the case of network display, an object from the visNetwork package.

Author(s)

Stefano M. Pagnotta

References

Cerulo, Pagnotta (2022) doi:10.3390/e24050739

See Also

massiveGST, visNetwork, visOptions

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")
```

```
# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

# to get the bar-plot
plot(cut_by_significance(ans, level_of_significance = 0.01))

# to get the network of the gene-sets
plot(cut_by_significance(ans, level_of_significance = 0.01),
     gene_sets = geneSets, as.network = TRUE)
```

`pre_ranked_list` *FGFR3-TACC3 fusion positive gene profile*

Description

This gene-profile comes from the paper in reference. It compares 9 FGFR3-TACC3 fusion positive samples versus 535 other samples in the GBM study from TCGA (Agilent platform).

Author(s)

Stefano M. Pagnotta

References

Frattini et al. "A metabolic function of FGFR3-TACC3 gene fusions in cancer" *Nature volume 553, 2018* doi:10.1038/nature25171

`save_as_tsv` *Save the results in tab-separeted value file*

Description

Save the data frame coming from the massiveGST function as tab-separeted value.

Usage

```
save_as_tsv(x, file_name = "massiveGST.tsv", sep = "\t", ...)
```

Arguments

`x`	a data frame of "mGST" class coming from massiveGST function.
`file_name`	a character value ("massiveGST.tsv" as default)
`sep`	a character value
`...`	Arguments to be passed to methods

Value

No return value.

Author(s)

Stefano M. Pagnotta

See Also

massiveGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

# save the results
fname <- file.path(tempdir(), "massiveGST_results.tsv")
save_as_tsv(ans, file_name = fname)
```

`save_as_xls` *Save the results in xls file format*

Description

Save the data frame coming from the massiveGST function as Excel 2003 (XLS) or Excel 2007 (XLSX) files

Usage

```
save_as_xls(x, file_name = "massiveGST.xls", ...)
```

Arguments

`x`	a data frame of "mGST" class coming from massiveGST function.
`file_name`	a character value ("massiveGST.xls" as default)
`...`	Arguments to be passed to methods

Value

No return value.

Author(s)

Stefano M. Pagnotta

See Also

WriteXLS, massiveGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

# save the results
fname <- file.path(tempdir(), "massiveGST_results.xls")
save_as_xls(ans, file_name = fname)
```

`summary.mGST` *Generate summary tables*

Description

This method handles the result of massiveGST function, to provide views of the table.

Usage

```
## S3 method for class 'mGST'
summary(object,
  cols_to_remove = "link",
 order_by = c("relevance", "NES", "logit2NES", "p.value", "BH.value", "bonferroni"),
  top = NULL,
  as.formattable = FALSE,
  ...
)
```

Arguments

`object`	a data structure coming from the massiveGST function
`cols_to_remove`	A character list of the columns to remove from the output.
`order_by`	a character string specifying which marginal ordering has to be applied to the output; must be one of "relevance" (default), "NES", "logit2NES", "p.value", "BH.value", and "bonferroni"
`top`	an integer to trim the table to the first 'top' rows.
`as.formattable`	a logical value (default = FALSE) to provide a formatted output with the help of formattable package.
`...`	Arguments to be passed to methods

Value

A data frame.

Author(s)

Stefano M. Pagnotta

See Also

massiveGST

Examples

```
library(massiveGST)

# get the gene profile
fname <- system.file("extdata", package="massiveGST")
fname <- file.path(fname, "pre_ranked_list.txt")
geneProfile <- get_geneProfile(fname)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# run the function
ans <- massiveGST(geneProfile, geneSets, alternative = "two.sided")

summary(ans)
summary(ans, as.formattable = TRUE, order_by = "NES", top = 10)
```

`write_geneSets_to_gmt` *Save a collection of gene-sets in a .gmt file format.*

Description

Write a collection of gene sets as arranged in this package in a gmt file format.

Usage

```
write_geneSets_to_gmt(gs, fileName)
```

Arguments

`gs`	a character vector of gene-sets
`fileName`	a character value; "gene_sets.gmt" (default)

Value

No return value.

Author(s)

Stefano M. Pagnotta

See Also

get_geneSets_from_msigdbr, get_geneSets_from_local_files

Examples

```
library(massiveGST)

# get the gene-sets
geneSets <- get_geneSets_from_msigdbr(category = "H", what = "gene_symbol")

# save the gene-sets
fname <- file.path(tempdir(), "hallmarks.gmt")
write_geneSets_to_gmt(geneSets, fileName = fname)
```

Index

cut_by_logit2NES, 2, *4*, *5*, *11*
cut_by_NES, *2*, 3, *5*, *11*
cut_by_significance, *2*, *4*, 4, *11*, *12*

dist, *6*

fisher.test, *12*

geneSets.sim, 6
get_geneProfile, 7
get_geneSets_from_local_files, 8, *19*
get_geneSets_from_msigdbr, *8*, 9, *19*

massiveGST, *2*, *4*, *5*, 10, *14*, *16–18*
massiveORT, 11
msigdbr, *9*

plot.mGST, *2*, *4*, *5*, *11*, 13
pre_ranked_list, *7*, 15

save_as_tsv, 15
save_as_xls, 16
summary.mGST, *2*, *4*, *5*, *11*, *13*, 17

visNetwork, *14*
visOptions, *14*

write_geneSets_to_gmt, *8*, 19
WriteXLS, *17*