GSEA.plot

Civelek Lab University of Virginia

Sarah Innis, Mete Civelek, Warren Anderson

August 29, 2019

This vignette gives an introduction into the GSEA.plot package. This package builds off of the Gene Set Enrichment computational analysis published by the Broad Institute specifically their R code for outputting the results. This package uses the enrichment data to create plots that are easier to understand and modify. In addition to improved plotting functionality, this package also introduces the capability to create your own geneset databases. The package includes all geneset databases published by the Broad Institute as well as 6 different transcription factor databases and two databases containing ligands and receptor pairs.

This package requires three data files. It will output two summary results files and a pdf of all of the enrichment plots to the current working directory. Within the current working directory a folder will be created with the name assigned to doc.string in the GSEAplots function call. The individual gene set results will be saved into this folder.

Load the Package

```
#devtools::install_github("sarah-innis/GSEA.plot")
library(devtools)
library(GSEA.plot)
```

Geneset Database Options

To improve on the GSEA code published by the Broad Institute, additional geneset database options were included within this package.

Existing Databases

Calling the key included within the package can show you which databases are currently saved in the package.
.

```
data(key)
head(key)
```

```
##                File                                 Description
## 1             C1.gs                            positional all
## 2          C2.cgp.gs curated chemical and genetic perturbations
## 3 C2.cp.biocarta.gs        curated canonical pathways biocarta
## 4           C2.cp.gs                 curated canonical pathways
## 5      C2.cp.kegg.gs            curated canonical pathways kegg
## 6 C2.cp.reactome.gs        curated canonical pathways reactome
##                                         Source
## 1 http://software.broadinstitute.org/gsea/msigdb
## 2 http://software.broadinstitute.org/gsea/msigdb
## 3 http://software.broadinstitute.org/gsea/msigdb
## 4 http://software.broadinstitute.org/gsea/msigdb
## 5 http://software.broadinstitute.org/gsea/msigdb
## 6 http://software.broadinstitute.org/gsea/msigdb
```

The key has three columns: file, description, and source. Each saved database has a more descriptive name and then the file name that must be used when it is called to be used. The purpose of this is to make the various databases more accessible. For instance, one is more likely to know they are looking for the Gene Ontology molecular function database then they are to know the name of the file that is needed to use the database. The function database_key allows you to find the file name for a given descriptive name. It also has the option to return the descriptive names of every available database when the function input is "all."

```
GO_mf_filename=database_key("GO molecular function")
GO_mf_filename
```

```
## [1] "C5.mf.gs"
```

```
descriptive_names=database_key("all")
head(descriptive_names)
```

```
## [1] positional all
## [2] curated chemical and genetic perturbations
## [3] curated canonical pathways biocarta
## [4] curated canonical pathways
## [5] curated canonical pathways kegg
## [6] curated canonical pathways reactome
## 68 Levels: computational all ... TRRUST transcription factors
```

Once you have found the filename for the database you are interested in, you can load it into your workspace and work with it.

```
data(hallmark.gs)
data(C1.gs)
```

To check if a geneset database includes the set you are interested in, you can view all sets within a database with the get_genesets function.

```
sets=get_genesets(hallmark.gs)
head(sets)
```

```
## [1] "HALLMARK_TNFA_SIGNALING_VIA_NFKB"
## [2] "HALLMARK_HYPOXIA"
## [3] "HALLMARK_CHOLESTEROL_HOMEOSTASIS"
## [4] "HALLMARK_MITOTIC_SPINDLE"
## [5] "HALLMARK_WNT_BETA_CATENIN_SIGNALING"
## [6] "HALLMARK_TGF_BETA_SIGNALING"
```

Setting up Data for the Package

3 dataframes are necessary to run the Gene Set Enrichment Analysis.You should have a geneset database, an expression file, and a phenotype file. You can write your own or use one of the included files. To load data included in the package use the data() function and then call the R objects. If you have not saved data to the package, set the object equal to the parameter.

Loading Data

```
data(aagmex_expr)
expr.input=aagmex_expr
expr.input[1:4,1:6]
```

```
##       GSM2520983 GSM2520984 GSM2520985 GSM2520986 GSM2520991 GSM2520992
## 7A5     6.416175   6.303225   6.292974   6.288139   6.354787   6.365153
```

```
## A1BG    6.442692   6.474838   6.538247   6.453630   6.498097   6.404469
## A1CF    6.429408   6.354601   6.380667   6.369513   6.386864   6.357330
## A26C3   6.431504   6.370737   6.325280   6.377389   6.376220   6.388600
```

```
data(aagmex_pheno)
pheno.input=aagmex_pheno
pheno.input$phen
```

```
## [1] "Female" "Male"
```

```
head(pheno.input$class.v)
```

```
## [1] 0 0 0 0 0 0
```

```
data(hallmark.gs)
gene.set.input=hallmark.gs
```

GSEA Analysis

The GSEA analysis takes three files. Instructions for how to prepare each file are included on the Broad Institute page for their GSEA analysis function. If instead of the preformatted data from the package you would like to use your own files, set the corresponding input to the file path. The expression data should be in .gct format, the phenotype in .cls format, and the gene set database in .gmt format.

Other Analysis Paramaters:

- nperm determines the number of permutations. Permutations are used to analyze the significance of the association between gene expression and phenotype and determine whether the associations are more significant then random ones.
- fdr.q.val.threshold sets a limit on false discovery rate. Set at a default of 0.25, we would expect that 3 out of 4 times the result is likely valid. The fdr value is computed as a ratio of the enrichment score to the enrichment scores for all permutations of the gene set over the ratio of the enrichment score to the enrichment scores of the actual gene set.
- abs.val is default false. When set to true genes are ranked according to the absolute value of their signal to noise ratio.
- gs.size.threshold.max This is the maximum number of gene symbols in expression data that match a given gene set. No more than this number will be analyzed.

Plotting Paramaters:

- bar_percent: the size of the enrichment tick mark relative to the size of the enrichment score plot
- gap_percent: the size of the gaph between the running enrichment score graph and the tick marks relative to the size of the enrichment score plot
- under_percent: the size of the space below the tick marks relative to the size of the enrichment score plot
- upper_percent: the size of the space above the enrichment plot relative to the size of the enrichment score plot
- color_line: color of the line in the enrichment score plot
- color_tick: color of the tick marks that indicate a gene symbol within a gene set that is differentially expressed in the data

The parameters gs.size.threshold.min and gs.size.threshold.max determines how many gene symbols from the gene set and the expression data can match. Their default values are respectively 25 and 1000. Gs.size.threshold.max can be changed within the GSEAplots() function.

```
pp= GSEAplots(input.ds.name=expr.input,
              input.cls.name=pheno.input, gene.set.input=gene.set.input,
              doc.string="Aagmex_hall", nperm=1000,
```

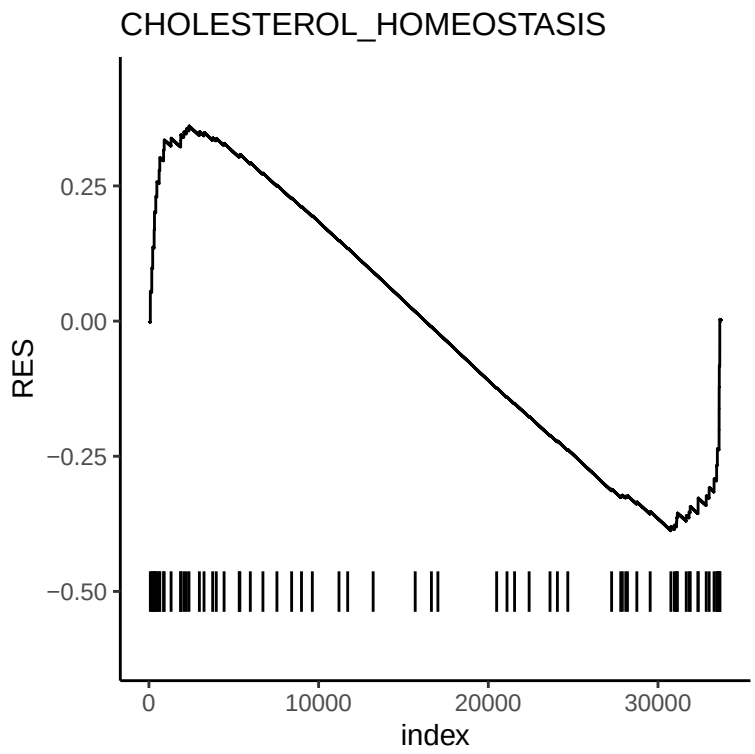
```
                fdr.q.val.threshold = 0.25,abs.val=F,gs.size.threshold.max=1000, bar_percent=0.1, gap_pe
                under_percent=0.1,upper_percent=0.1,color_line="black",
                color_tick="black")
```

Function outputs

Plots

This is the enrichment plot given from the first geneset.

```
pp$plots[[1]]
```



Use the function plot.ES to save a pdf containing the enrichment plots for every set in the gene set database.

```
plot.ES(list.of.plots=pp$plots,plotname="GSEA_plots")
```

Gene.Set.Reference.Matrix and Gene.set.leading

These are the gene symbols contained in the first gene set.

```
names(pp$gene.set.reference.matrix)[[1]]
```

```
## [1] "HALLMARK_TNFA_SIGNALING_VIA_NFKB"
```

```
head(pp$gene.set.reference.matrix[[1]])
```

```
## [1] "JUNB"    "CXCL2"   "ATF3"    "NFKBIA"  "TNFAIP3" "PTGS2"
```

These are the leading gene symbols for the first gene set.

```
names(pp$gene.set.reference.matrix)[[1]]
```

```
## [1] "HALLMARK_TNFA_SIGNALING_VIA_NFKB"
```

```
head(pp$gene.set.leading[[1]])
```

```
## [1] "FASN"   "ATF5"   "TM7SF2" "FABP5"  "CLU"    "LDLR"
```

Reports

One report is generated for each phenotype. The table shows all of the information available in the report file except for the source for the geneset which is ommitted for easier viewing.

```
pp$report1[1,-3]
```

```
##                              GS SIZE      ES    NES NOM p-val FDR q-val
## 1 HALLMARK_PANCREAS_BETA_CELLS   38 0.47384 1.7127  0.001883  0.014646
##   FWER p-val Tag % Gene % Signal FDR (median) glob.p.val
## 1      0.014 0.263  0.161  0.221            0      0.006
```

Enrichment Scores

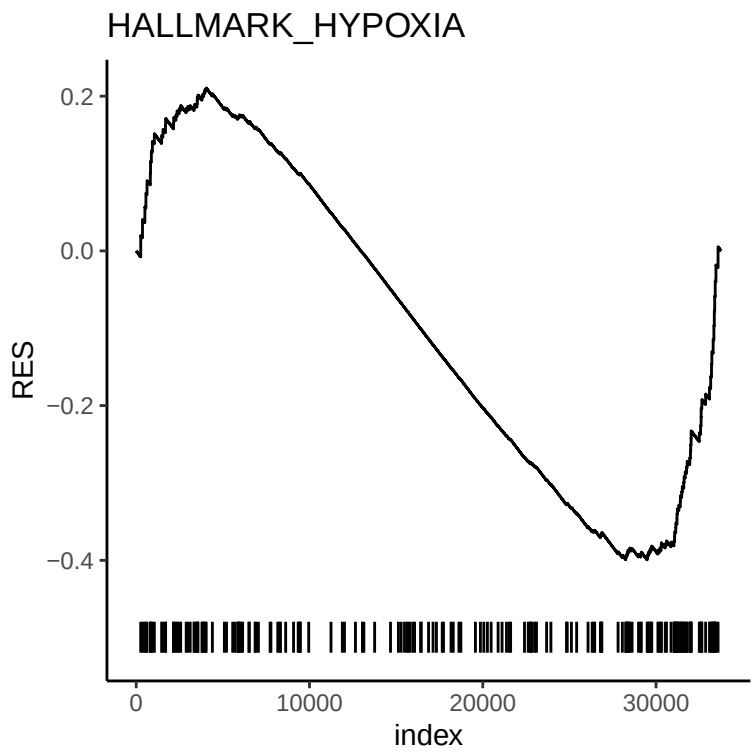
The enrichment score ouput allows you to access the data used to generate the plots so that you might create your own.

```
data1=pp$ES[[2]]
head(data1)
```

```
##   index           RES EStag
## 1     1 -2.978673e-05     0
## 2     2 -5.957345e-05     0
## 3     3 -8.936018e-05     0
## 4     4 -1.191469e-04     0
## 5     5 -1.489336e-04     0
## 6     6 -1.787204e-04     0
```

Enrichment scores are plotted for each position within the gene set. A positive enrichment score indicates correlation with the first phenotype which in the case of the aagmex pheno data is Female and a negative enrichment score indicates correlation with the second phenotype, Male. The gene symbols within your gene set that are differentially expressed are indicated with a tick mark under this plot.In the enrichment score data these symbols are indicated with an EStag of 1. To specify the dimensions and positions of the tick marks a data frame is created to create line segments that have a regular height and are positioned at each position with an EStag of 1. The segments should be vertical so the vx or change in x over the segment should be 0. The y position in the data frame dictates the starting y position of the mark. It is currently set to 0.12 below the minimum of of the enrichment plot. vy dictates the end of each mark to be 0.04 higher than that starting position. In the plotting function, 0.12 and 0.04 are variables that can be modified by the bar_percent and under_percent parameters. Using ggplot, the enrichment score plot and tick marks are combined into one plot called p.

```
enrich_ind=which(data1$EStag==1)
d=data.frame(x=enrich_ind, y=matrix(min(data1$RES)-0.12,length(enrich_ind),1),
             vx=matrix(0,length(enrich_ind),1), vy=matrix(0.04,length(enrich_ind),1))
p <- ggplot(data1, aes(index,RES))+geom_line(col="black")
p <- p+geom_segment(data=d, mapping=aes(x=x, y=y, xend=x+vx, yend=y+vy),col="black")
p <- p+theme_classic()
p <- p+ggtitle(names(pp$gene.set.reference.matrix)[[2]])
print(p)
```



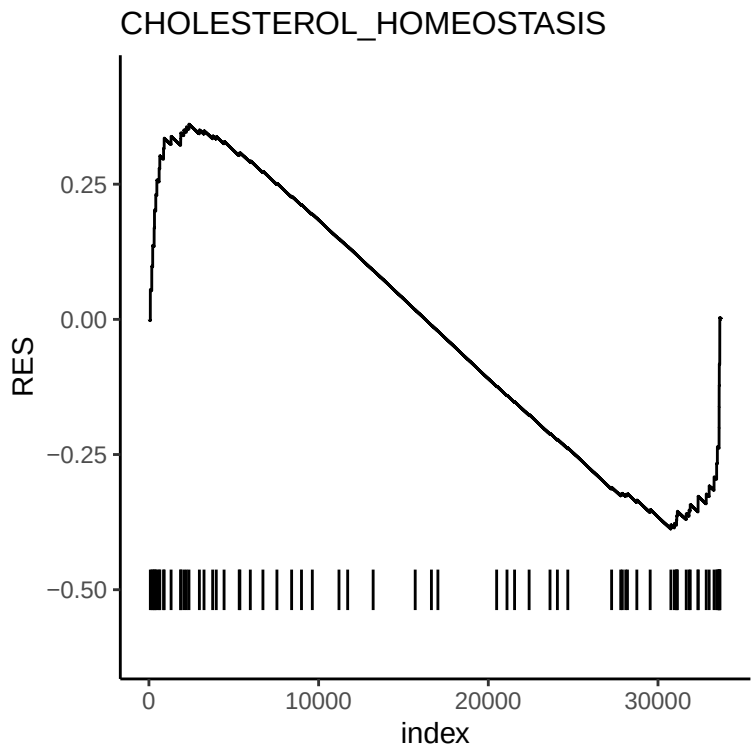
Running GSEA on a New Database

This package introduces the ability to create your own database. To create your own database you need to create a list. The names of the list must be the names of the genesets. The items in each element of the list are the description and gene symbols. For example the names of a potential list would be the transcription factors and the elements would be the genes affected by that specific transcription factor. The create_geneset_db() function then converts this list into a format that can go into the GSEAplots function.

```
sets=get_genesets(hallmark.gs)
symbols=get_genesymbols(hallmark.gs)
entry1=c("source_1",symbols$HALLMARK_TNFA_SIGNALING_VIA_NFKB)
entry2=c("source_2",symbols$HALLMARK_HYPOXIA)
entry3=c("source_3",symbols$HALLMARK_CHOLESTEROL_HOMEOSTASIS)
hall_13.db=list(entry1,entry2,entry3)
names(hall_13.db)=c(sets[1],sets[2],sets[3])
gene.set.input=create_geneset_db(hall_13.db)
```

```
hall_13_results= GSEAplots(input.ds.name=expr.input,
               input.cls.name=pheno.input, gene.set.input=gene.set.input,
               doc.string="Aagmex_hall13", nperm=1000,gs.size.threshold.max = 1000,
               fdr.q.val.threshold = 0.25, bar_percent=0.1, gap_percent=0.1,
               under_percent=0.1,upper_percent=0.1,color_line="black",
               color_tick="black",abs.val=F)
```

```
hall_13_results$plots[[2]]
```



Introduction of Transcription Factor Databases

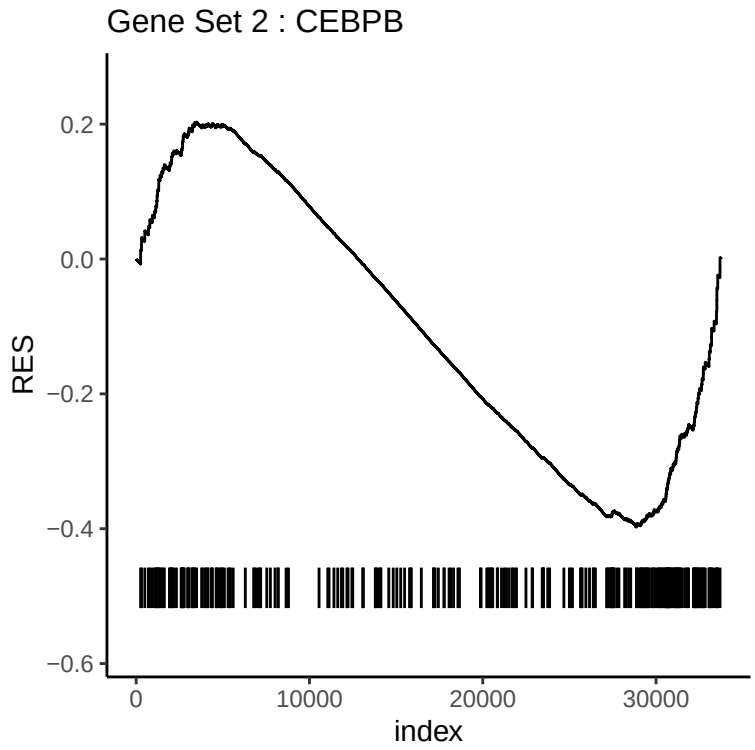
The following transcription factor databases were introduced to the Broad Institute gene set databases: Neph2012, ENCODE, ITFP, Marbach2016, TRRUST, and TRED. These databases were downloaded from the tfttargets package from https://github.com/slowkow/tftargets and reformatted for use in the GSEA function. The sets for these include the transcription factor name and then enrichment is obtained for the gene symbols affected by that transcription factor. This allows for expanded use to show which transcription factors and genes regulated by those correspond to expression data.

```
data(ENCODE.db)
gene.set.input=ENCODE.db

enrichment_TF= GSEAplots(input.ds.name=expr.input,
               input.cls.name=pheno.input, gene.set.input=gene.set.input,
               doc.string="Aagmex_TF", nperm=1000,gs.size.threshold.max=1000,fdr.q.val.threshold = 0.25
               bar_percent=0.1, gap_percent=0.1, under_percent=0.1,upper_percent=0.1,color_line="black"
               color_tick="black",abs.val=F)
```

```
 enrichment_TF$plots[1]
```

```
## [[1]]
```



Add to an Existing Gene Set Database

This new function allows you to add gene sets into an exsiting database. The database input needs to be already in the tab separated format that is used in the GSEA function. The gene sets to add have a separate line for each gene set with the gene set name, source, and gene symbols.

```
data(transf)
data(transm)
tx = t(c("HALLMARK_KLF14","http://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_KLF14",transf[,1],
```

The add_to_database function adds this set to the hallmark sets and formats it for use in the function. An example of it being used is in the section below.

```
gene.set.input=add_to_database(database=hallmark.gs,addition=tx)
head(gene.set.input)
```

```
## [1] "HALLMARK_TNFA_SIGNALING_VIA_NFKB\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_TNFA
## [2] "HALLMARK_HYPOXIA\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_HYPOXIA\tPGK1\tPDK1\
## [3] "HALLMARK_CHOLESTEROL_HOMEOSTASIS\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_CHOL
## [4] "HALLMARK_MITOTIC_SPINDLE\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_MITOTIC_SPIN
## [5] "HALLMARK_WNT_BETA_CATENIN_SIGNALING\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_W
## [6] "HALLMARK_TGF_BETA_SIGNALING\thttp://www.broadinstitute.org/gsea/msigdb/cards/HALLMARK_TGF_BETA_
```

Create a Phenotype Input File

The pheno.input parameter for GSEA.plot requires a list with a phen element and a class.v element.

```
data(aagmex_pheno)
aagmex_pheno$phen
```

```
## [1] "Female" "Male"
```

```
head(aagmex_pheno$class.v)
```

```
## [1] 0 0 0 0 0 0
```

The create_phenoinput file takes a character column with labels for the phenotype of each sample and makes it into a working pheno.input for the GSEA.plot function.

```
data(gtex_ann)
gtex_ann
```

```
##   [1] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##   [8] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [15] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [22] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [29] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [36] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [43] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [50] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [57] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [64] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [71] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [78] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [85] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [92] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
##  [99] "Female" "Female" "Female" "Female" "Female" "Female" "Female"
## [106] "Female" "Female" "Female" "Female" "Female" "Female" "Male"
## [113] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [120] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [127] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [134] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [141] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [148] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [155] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [162] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [169] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [176] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [183] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [190] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [197] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [204] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [211] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [218] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [225] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [232] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [239] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [246] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [253] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [260] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [267] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [274] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [281] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [288] "Male"   "Male"   "Male"   "Male"   "Male"   "Male"   "Male"
## [295] "Male"   "Male"   "Male"   "Male"
```

```
pheno.input=create_phenoinput(gtex_ann)
pheno.input$phen
```

```
## [1] "Female" "Male"
```

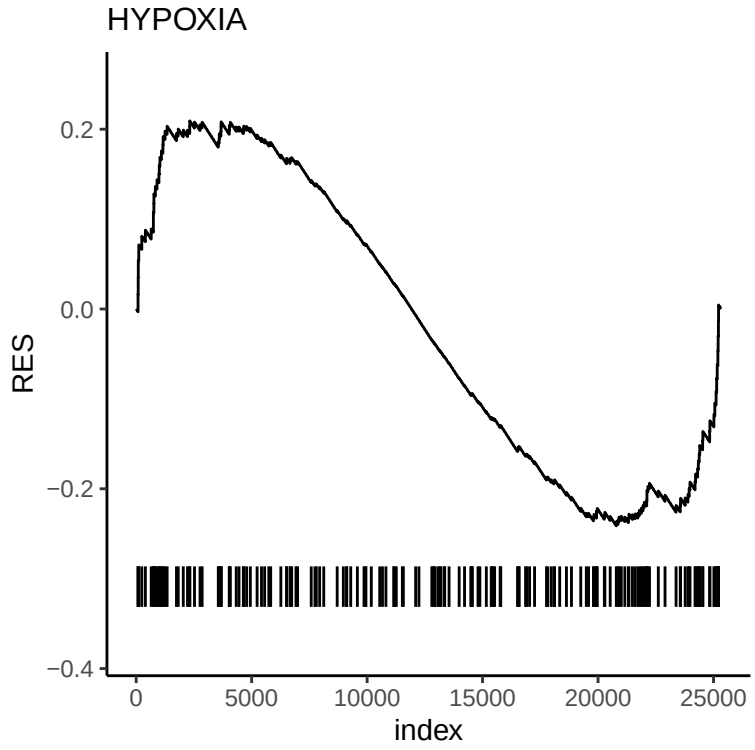
```
head(pheno.input$class.v)
```

```
## [1] 0 0 0 0 0 0
```

```
data(gtex_expr)
expr.input=gtex_expr
```

```
gtex_example= GSEAplots(input.ds.name=expr.input,
                input.cls.name=pheno.input, gene.set.input=gene.set.input,
                doc.string="Gtex_hall", nperm=1000,gs.size.threshold.max = 1000,
                fdr.q.val.threshold = 0.25,abs.val=F,bar_percent=0.1, gap_percent=0.1,
                under_percent=0.1,upper_percent=0.1,color_line="black",
                color_tick="black")
```

```
gtex_example$plots[[2]]
```



References

- Subramanian, Tamayo, et al. (2005), PNAS 102, 15545-15550, http://www.broad.mit.edu/gsea/
- https://github.com/slowkow/tftargets
- https://www.ncbi.nlm.nih.gov/pubmed/28942919