GSEAplot - R package

Civelek Lab, University of Virginia

Sarah Innis, Kelsie Reinaltt, Mete Civelek, Warren Anderson

July 13, 2020

Contents

Introduction to GSEAplot

This vignette provides an introduction to the GSEAplot package. This package implements the Gene Set Enrichment Analysis (GSEA), a computational method published by the Broad Institute (Subramanian et al., 2005). This package uses the enrichment data from the GSEA analysis to create customizable plots and facilitates the analysis of customized gene set libraries. The package includes all gene set databases published by the Broad Institute as well as 6 different transcription factor databases and two databases containing ligands and receptor pairs.

Three data objects are needed in order to perform the gene set enrichment analysis: a gene expression data set, a phenotype annotation, and a gene set library. The **GSEAplots** function executes the analysis and produces a summary results file for each phenotype value in the 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 results for each gene set within the database will be saved into this folder. The **plot.ES** function saves a pdf of all generated enrichment graphs into the working directory.

Loading the Package

The following script will install and load the package:

```
remotes::install_github("kelsiereinaltt/GSEAplot")
library(GSEAplot)
```

Geneset Database Options

This package includes additional gene set database options. The **GSEAplots** function was made to be compatible with custom gene set libraries or existing gene set libraries modified by the user.

Identifying and Loading Existing Databases

Calling the key included within the package shows which databases (gene set libraries) 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. The descripton correspoinding to a file provides an intuitive description of the database contents. Users can identify a gene set library of interest by reading its description then loading the database using its file name.

database_key can also return the descriptons for all databases when the function input is "all". Given the descriptions of all databases, the user may select the file of interest.

```
descriptive_names=database_key("all")
descriptive_names[20]
```

```
## [1] "hallmark all"
```

```
key[20,]
```

```
##           File  Description                                          Source
## 20 hallmark.gs hallmark all http://software.broadinstitute.org/gsea/msigdb
```

The function **database_key** returns the file name of a database given the description of the database.

```
filename=database_key("hallmark all")
filename
```

```
## [1] "hallmark.gs"
```

Once the file name is identified, the database of interest may be loaded into the workspace.

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

To check if a geneset database includes gene sets with functional annotations of interest, all sets within a database can be viewed using the **get_genesets** function.

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

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

Setting up Data for Gene Set Enrichment Analysis using GSEAplot

Gene Set Enrichment Analysis requires the user to provide three data objects: a gene expression data frame, a phenotype list, and a gene set library formatted as a character vector with one entry for each gene set. The user may use sample data provided in this package, or format their own data following the examples below. To load the sample data included in the package, call the R dataset with the **data()** function.

The package includes publically available human subcutaneous adipose tissue gene expression data from males and females of the African American Genetics of Metabolism and Expression cohort (AAGMEx, Sharma et al., 2016) and the Genotype-Tissue Expression cohort (GTEx, GTEx Consortium et al., 2017).

Formatting Data

The parameter *input.ds.name* of the **GSEAplots** function requires *gene expression data with two phenotypes* for a comparison of interest. The following is an example of how gene expression data included this package can be organized:

```
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
```

The parameter *input.cls.name* of the **GSEAplots** function takes the *phenotype* data, which maps the differential expression data to one of the two phenotypes. The following is an example of how the phenotype data included this package can be organized:

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

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

```
head(pheno.input)
```

```
## $phen
## [1] "Female" "Male"
##
## $class.v
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

The parameter gene.set.input takes a *geneset* or a geneset database (collection of genesets). The following is an example of how geneset data included this package can be loaded:

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

Custom phenotype inputs can be created as follows. The phenotype input consists of 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** function takes a character vector with labels for the phenotype of each sample and makes it into a correctly formatted phenotype input list for the **GSEAplots** function.

```
data(gtex_ann)
head(gtex_ann)
```

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

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

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

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

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

The parameter *input.ds.name* of the **GSEAplots** function takes a *gene set library* (collection of gene sets) for the analysis. The following is an example of how gene set data included this package can be loaded:

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

GSEA Analysis

GSEAplots: Function and Parameters

In addition to the three data objects, the GSEA analysis offers a number of other parameters.

Analysis and Input Paramaters:

- *input.ds.name* - gene expression data frame
- *input.cls.name* - phenotype input list
- *gene.set.input* - geneset database
- *doc.string* - name of the folder where GSEA results are saved to
- *nperm* - the number of permutations used to determine the significance of the association between a geneset annotation and the phenotype

• *abs.val* - when set to true (T), genes are ranked according to the absolute value of their signal to noise ratio (default = FALSE)

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 gap 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 genes in a given gene set

The following script is an example of how to use the **GSEAplots** function. The run time for this function with its current parameters is approximately 3.2 minutes on a computer with a MacOS 10.14 (Mojave) operating system, 2.2 GHz Intel Core i7 processor, and 16 GB of memory.

```
start_time <-Sys.time()

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

 end_time <- Sys.time()
 end_time-start_time
```

Analysis Outputs

Outputs of **GSEAplots** include reports, enrichment scores, plots, and the objects **gene.set.reference.matrix** and **gene.set.leading**. All outputs will be saved to the working directory.

Reports

A report is generated for each phenotype (output tables include *report1* and *report2*). An example of the data is provided below. In this case, the first row contains the data for the geneset HALLMARK_PANCREAS_BETA_CELLS. The third column is the source which is ommitted for streamlined viewing.

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

```
##                                    GS SIZE      ES    NES NOM p-val FDR q-val
## 1 HALLMARK_OXIDATIVE_PHOSPHORYLATION  187 0.70426 2.3103         0         0
##   FWER p-val Tag % Gene % Signal FDR (median) glob.p.val
## 1          0 0.561  0.135  0.489            0          0
```

Enrichment Scores

The list *ES* provides access to the enrichment score data used to generate the enrichment plots along with the corresponding gene symbols.

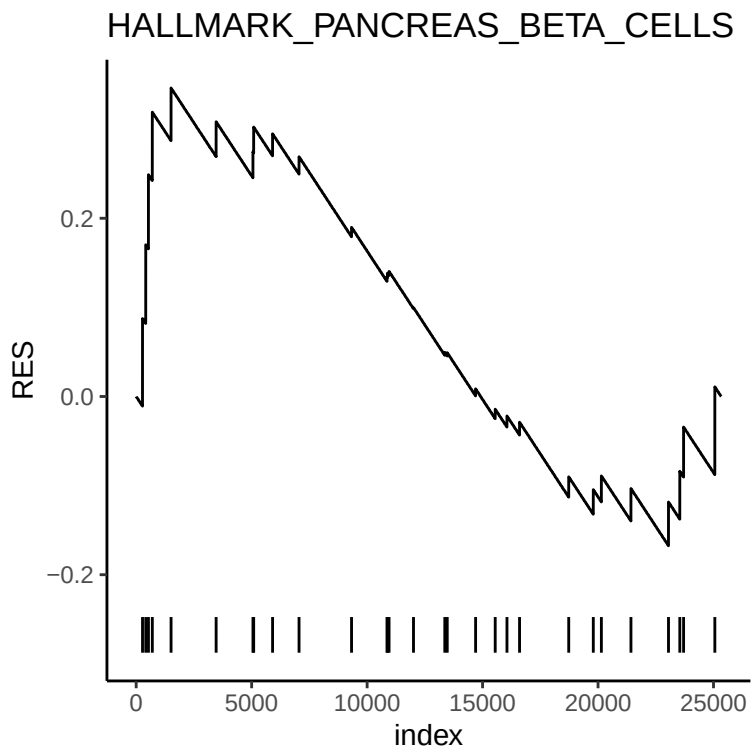
```
data1=pp$ES[[50]]
head(data1)
```

```
##   index gene.symbol           RES EStag
## 1     1    EIF1AXP1 -3.952101e-05     0
## 2     2     EIF2S3L -7.904201e-05     0
## 3     3    ANKRD30A -1.185630e-04     0
## 4     4        SAA1 -1.580840e-04     0
## 5     5    ATP8A2P1 -1.976050e-04     0
## 6     6   LINC00993 -2.371260e-04     0
```

Access to enrichment score data allows for further analysis. For instance, users may create their own custom plots. The following describes a sample procedure for producing custom plots.

Enrichment scores are plotted for each position within the gene set. A positive enrichment score indicates correlation with the first phenotype value (e.g., Female) and a negative enrichment score indicates correlation with the second phenotype value (e.g., Male). The gene symbols within your gene set that are ranked according to differential expression are indicated with a tick mark under this plot. The enrichment score data for these symbols are indicated with an EStag of 1. To specify the dimensions and positions of the tick marks, a data frame can be generated to create line segments that have a regular height and are positioned corresponding to EStag values 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. The vy parameter dictates the end of each mark, which is set to be 0.04 higher than the 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 can be combined into one plot.

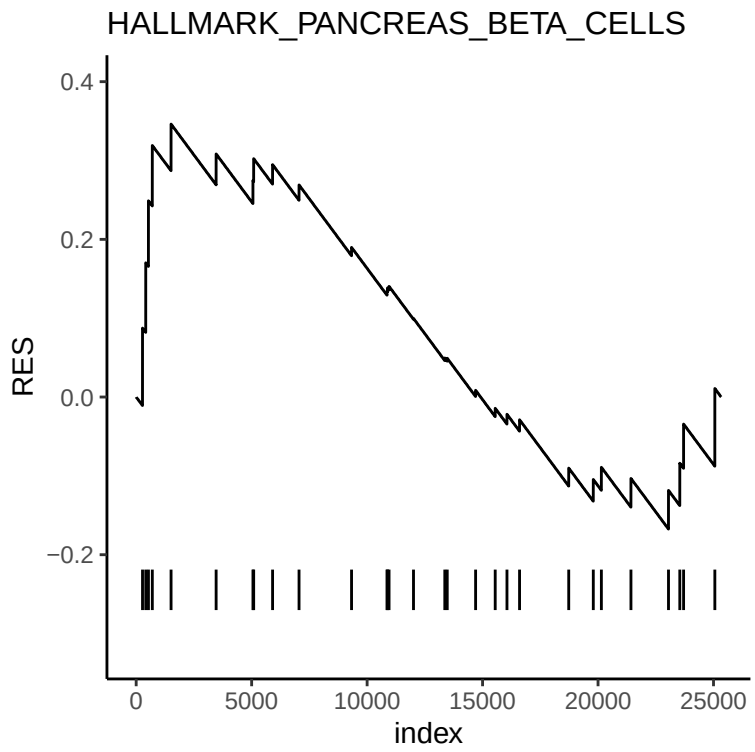
```
enrich_ind=which(data1$EStag==1)
d=data.frame(x=enrich_ind, y=matrix(min(data1$RES)-0.12,length(enrich_ind),1),
             vx=matrix(0,length(enrich_ind),1), vy=matrix(0.04,length(enrich_ind),1))
p <- ggplot(data1, aes(index,RES))+geom_line(col="black")
p <- p+geom_segment(data=d, mapping=aes(x=x, y=y, xend=x+vx, yend=y+vy),col="black")
p <- p+theme_classic()
p <- p+ggtitle(names(pp$gene.set.reference.matrix)[[50]])
print(p)
```



Plots

The following is an example of an erichment plot for the first gene set:

```
pp$plots[[50]]
```



Use the function **plot.ES** to save a pdf of enrichment plots for all sets in the gene set database. The pdf will be written to the working directory.

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

The *gene.set.reference.matrix* and *gene.set.leading* outputs

gene.set.reference.matrix is a table of the gene sets being studied.

```
names(pp$gene.set.reference.matrix)[[50]]
```

```
## [1] "HALLMARK_PANCREAS_BETA_CELLS"
```

```
head(pp$gene.set.reference.matrix[[50]])
```

```
## [1] "PAX6"    "NEUROD1" "ISL1"    "NKX2-2"  "PCSK1"   "NKX6-1"
```

gene.set.leading is a list containing the leading edge set (set of genes with a degree of differential expression that is most related to the phenotype) for each gene set. If a sepecific gene set is more strongly related to the first phenotype value (e.g., Female), then the maximum absolute value of the runnning enrichment score (RES) will be positive. In this case, the leading edge set is all of the annotated genes that are to the left of and including the maximal enrichment score. If a given gene set is more related to the second phenotype value (e.g., Male), then the largest running enrichment score will be negative. In this case, the leading edge set is all of the annotated genes that are to the right of and including the maximal enrichment score.

The following example displays the leading gene symbols for one gene set:

```
names(pp$gene.set.reference.matrix)[[50]]
```

```
## [1] "HALLMARK_PANCREAS_BETA_CELLS"
```

```
head(pp$gene.set.leading[[50]])
```

```
## [1] "VDR"   "SRPRB" "AKT3"  "FOXO1"
```

Running GSEA for a Novel Gene sets

This package has functionality for using custom geneset libraries. For example, the user can include genes regulated by a specific transcription factor.

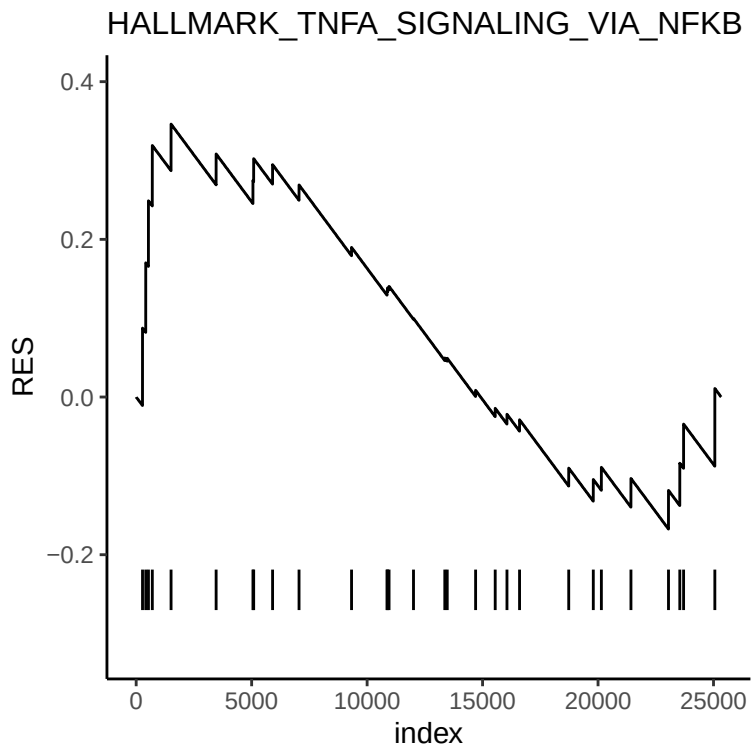
Generate a Novel Geneset Library

The **create_geneset_db** function converts a gene list into a format compatible with the **GSEAplots** function. The following example illustrates how to create a custom geneset database for **GSEAplots**:

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

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

```
custom_results$plots[[1]]
```



Add to an Existing Gene Set Library

The **add_to_database()** function allows the user to add gene sets into an existing library. These sets can be converted into the correct format with the **add_to_database()** function. For example, the user can add a gene set with putative targets of the transcription factor KLF14 (Small et al., 2018; Civelek et al., 2017). This requires the user to create a vector with the name of the set, the source, and then the gene symbols.

```
data(transf)
data(transm)
tx = t(c("KLF14_targets","custom",transf[,1], transm[,1]))
```

The **add_to_database** function can add this gene set to the hallmark gene sets and format it for use in the **GSEAplots** function.

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

Gene Sets Included in the Package

Ligand and Receptor Gene Sets

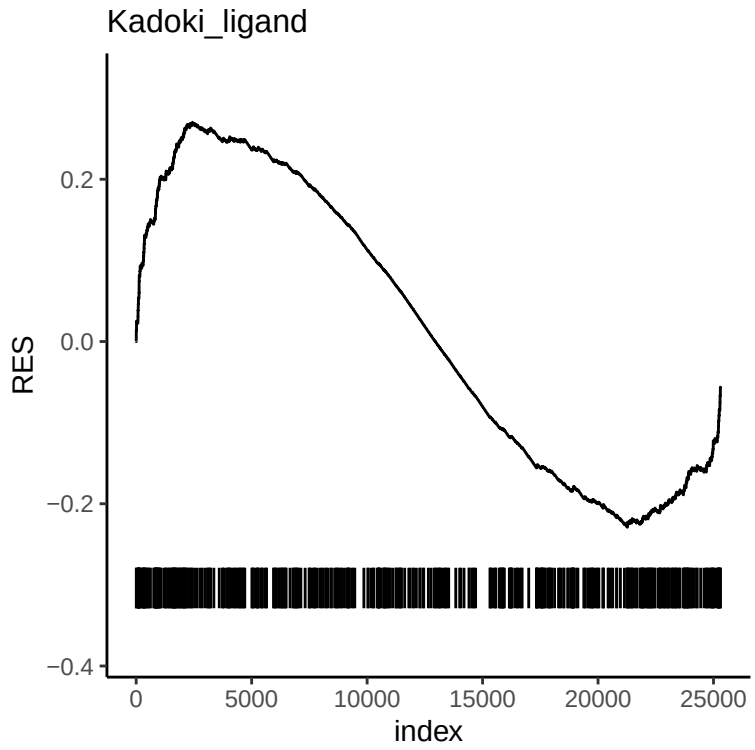
This package contains ligand and receptor gene sets based on curated lists of the respective annotations (Kadoki et al., 2017).

```
data(Kadoki_ligands.db)
data(Kadoki_receptors.db)
custom_db=c(Kadoki_ligands.db,Kadoki_receptors.db)
names(custom_db) = c("ligands", "receptors")
gene.set.input=custom_db

ligand_results=GSEAplots(input.ds.name=expr.input,
                         input.cls.name=pheno.input,
                         gene.set.input=gene.set.input,
                         doc.string="Aagmex_ligands",
                         nperm=1000,
                         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)
```

```
ligand_results$plots[1]
```

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



Transcription Factor Target Gene Sets

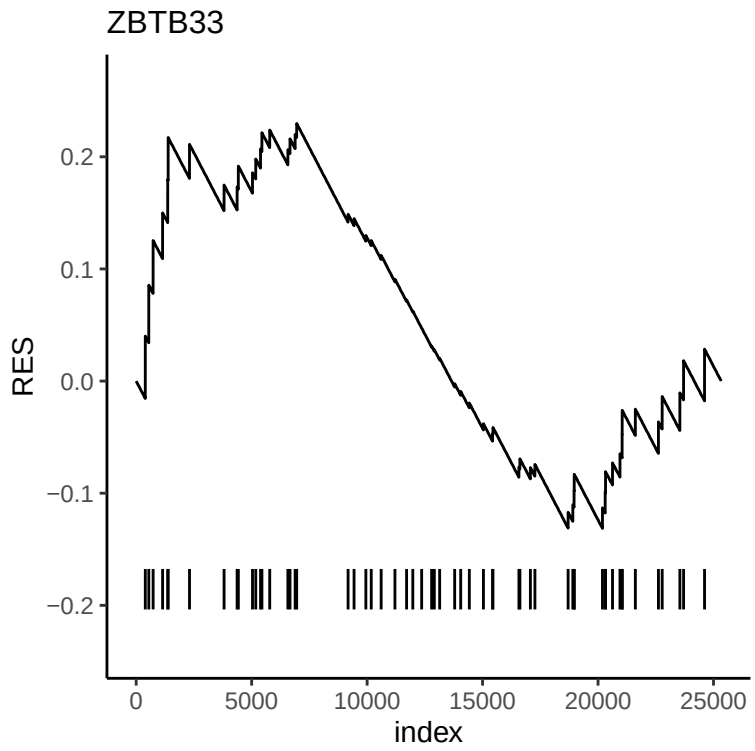
The following transcription factor databases were included (originally from the TFTargets package): Neph2012, ENCODE, ITFP, Marbach2016, TRRUST, and TRED (https://github.com/slowkow/tftargets).

```
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,
                         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]]
```



References

- Civelek et al. (2017). American Journal of Human Genetics 100, 428-443.
- GTEx Consortium et al. (2017). Nature 550, 204-213.
- Kadoki et al. (2017). Cell 171, 398-413.
- Sharma et al. (2016). The Journal of Clinical Endocrinology and Metabolism 101, 1455–1468.

- Small et al. (2018). Nature Genetics 50, 572-580.
- Subramanian et al. (2005). PNAS 102, 15545-15550. http://www.broad.mit.edu/gsea/
- https://github.com/slowkow/tftargets