Gene Set Enrichment Meta-Analysis with GSEMA package

Juan Antonio Villatoro-García [1,2] ***and Pedro Carmona-Sáez*** [*]
[1,2]

[1]Department of Statistics and Operational Research. University of Granada
[2]GENYO, Centre for Genomics and Oncological Research

[*]pedro.carmona@genyo.es

April 17, 2024

Abstract

GSEMA (Gene Set Enrichment Meta-Analysis) performs all the necessary steps of a gene set enrichment meta-analysis based on the combination of effect size. In preparation for calculating different effect sizes, first, a single sample enrichment analysis is performed. Based on the various enrichment values, the effect size is calculated as a difference of means. Subsequently, the package allows for the common steps of a meta-analysis based on the effect sizes.

packageVersionGSEMA 0.99.0

Contents

1 Introduction

GSEMA is a package designed to perform gene set enrichment meta-analysis. The gene set enrichment meta-analysis allows for obtaining the differentially regulated gene sets or pathways that are shared across various studies. Specifically, GSEMA applies a meta-analysis based on the combination of effect sizes obtained from single sample enrichment (SSE) analysis applied to individual the different studies. The different effect sizes of each study for each of the gene sets are calculated using a difference of means based on the enrichment scores obtained from SSE analysis. GSEMA offers various methods to obtain the statistics for the difference of means.

Subsequently, GSEMA allows for applying the various steps typical of a meta-analysis, in a similar way to the gene expression meta-analysis [1], although different adjustments to the calculated statistics are performed in order to correct the possible existence of false positive bias.

This vignette provides a tutorial-style introduction to all the steps necessary to properly conduct gene set enrichment meta-analysis using the **GSEMA** package.

2 Previous steps: Meta-analysis object

GSEMA uses a specific object as input, which is a list of nested lists where each nested list corresponds to a study. This object can be created directly by the users or they can use *createObjectMApath()* function to create it.

For the examples that are going to be shown, synthetic data will be used. We load the sample data into our R session.

```
> library(GSEMA)
> data("simulatedData")
```

- study1Ex, study2Ex, study3Ex, study4Ex, study5Ex: five expression matrices.
- study1Pheno, study2Pheno, study3Pheno, study4Pheno, study5Pheno: five phenodata objects.
- GeneSets: a list of gene sets with each element are the genes that belong to a pathway.
- objectMApathSim: the meta-analysis object created from the different expression matrices and phenodatas.

2.1 Meta-analysis object creation (objectMApath)

As previously stated, the meta-analysis input in GSEMA is a list of nested lists. Each nested list contains two elements:

- A gene set matrix with gene sets in rows and samples in columns
- A vector of 0 and 1 indicating the group of each sample. 0 represents reference group (usually controls) and 1 represents experimental group (usually cases).

This object can be created directly by the user or we can make use of *createObjectMApath()* function, which creates the *objectMApath* after indicating:

- the reference group (refGroups) and the experimental group (expGroups) in the phenodata.
- the gene sets (geneSets) that will be consider.
- the single sample enrichment method to applied to calculate the gene sets matrices

createObjectMApath() function allows to create the object needed to perform meta-analysis. In this case, it is necessary to indicate as input of the function the variables that contain the experimental and reference groups:

- listEX: a list of dataframes or matrix (genes in rows and samples in columns). A list of ExpressionSets can be used too:

```
> #List of expression matrices
> listMatrices <- list(study1Ex, study2Ex, study3Ex, study4Ex, study5Ex)
```

- listPheno: a list of phenodatas (samples in rows and covariables in columns). If the object listEX is a list of ExpressionSets this element can be null.

```
> listPhenodata <- list(study1Pheno, study2Pheno, study3Pheno, study4Pheno,
+     study5Pheno)
```

- namePheno: a list or vector of the different column names or column positions from the pheno used for perfoming the comparison between groups. Each element of namePheno correspont to its equivalent element in the listPheno. (default a vector of 1, all the first columns of each elements of listPheno are selected)
- expGroups: a list or vector of the group names or positions from namePheno variable used as experimental group (cases) to perform the comparison (default a vector of 1, all the first groups are selected).
- refGroups: a list or vector of the group names or positions from namePheno variable used as reference group (controls) to perform the comparison (default a vector of 2, all the second groups are selected).

  It is important to note that if any element does not belong to the experimental or the reference group, that sample is not taken into account in the creation of meta-analysis object.

  Here, we have included an example to show how exactly the function is used:

  Since this function can be a bit complicated if there are many datasets, we recommend creating a vector to keep the column names of the phenodatas that contains the variable that identifies the groups to compare (*namePheno* argument). Moreover, we should create two others lits to indicate how to identify experimental (cases) and reference (controls) groups in these variables (*expGroups* and *refGroups* arguments).

  If we look at the example phenodatas we can observed that the groups variable is "condition". Experimental group is named as "Case" and reference group as "Healthy".

```
> study1Pheno$Condition

 [1] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
 [8] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
[15] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
[22] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
[29] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
```

```
 [36] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
 [43] "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy" "Healthy"
 [50] "Healthy" "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [57] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [64] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [71] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [78] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [85] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [92] "Case"    "Case"    "Case"    "Case"    "Case"    "Case"    "Case"
 [99] "Case"    "Case"
```

- geneSets: a list of gene sets with each element are the genes that belong to a pathway:

```
> head(names(GeneSets))

[1] "SA_B_CELL_RECEPTOR_COMPLEXES" "SA_CASPASE_CASCADE"
[3] "SA_FAS_SIGNALING"             "SA_G1_AND_S_PHASES"
[5] "SA_G2_AND_M_PHASES"           "SA_MMP_CYTOKINE_CONNECTION"

> GeneSets[[1]]

 [1] "ATF2"     "BCR"      "BLNK"     "ELK1"     "FOS"      "GRB2"
 [7] "HRAS"     "JUN"      "LYN"      "MAP2K1"   "MAP3K1"   "MAPK1"
[13] "MAPK3"    "MAPK8IP3" "PAPPA"    "RAC1"     "RPS6KA1"  "RPS6KA3"
[19] "SHC1"     "SOS1"     "SYK"      "VAV1"     "VAV2"     "VAV3"
```

- pathMethod: a character string indicating the method to calculate the gene sets matrices. The available methods are:
  - "GSVA": Gene Set Variation method [2]
  - "Zscore": Z-score method [3]
  - "ssGSEA": Single Sample Gene Set Enrichment Analysis method [4]
  - "Singscore": Single sample scoring of molecular phenotypes [5]
- minSize: Minimum size of the resulting gene sets after gene identifier mapping. By default, the minimum size is 7.
- kdcf: Only neccesary for the GSVA method. Character vector of length 1 denoting the kernel to use during the non -parametric estimation of the cumulative distribution function of expression levels across samples. By default, kcdf="Gaussian" which is suitable when input expression values are continuous, such as microarray. When input expression values are integer counts, such as those derived from RNA-seq experiments, then this argument should be set to kcdf="Poisson".
- normalize: boolean specifying if the gen set matrices should be normalized. Default value "TRUE".
- n.cores: Number of cores to use in the parallelization of the datsets. By default, n.cores=1.
- internal.n.cores: Number of cores to use in the parallelization of the single sample enrichment methods. By default internal.n.cores= 1.

In parallelization, several aspects must be considered. "n.cores" refers to the parallelization of studies or datasets. Therefore, if we have 3 studies, the maximum number for "n.cores" will be 3. internal.n.cores refers to the parallelization of single sample enrichment methods. This is especially recommended for the ssGSEA method. For Singscore and GSVA, it may also be advisable. The process is parallelized based on the samples in each study. Therefore, the larger the number of samples, the slower the process will be. The number of cores that the computer will use is the multiplication of both parameters n.cores * internal.n.cores = total cores.

We all this information we can create the vector for *namePheno* argument and the two list for *expGroups* and *refGroups*:

```
> listMatrices <- list(study1Ex, study2Ex, study3Ex, study4Ex, study5Ex)
> listPhenodata <- list(study1Pheno, study2Pheno, study3Pheno, study4Pheno,
+     study5Pheno)
> phenoGroups <- c("Condition","Condition", "Condition", "Condition", "Condition")
> phenoCases <- list("Case", "Case", "Case", "Case", "Case")
> phenoControls <- list("Healthy", "Healthy", "Healthy", "Healthy", "Healthy")
> objectMApathSim <- createObjectMApath(
+     listEX = listMatrices,
+     listPheno = listPhenodata, namePheno = phenoGroups,
+     expGroups = phenoCases, refGroups = phenoControls,
+     geneSets = GeneSets,
+     pathMethod = "Zscore",
+     n.cores = 1,
+     internal.n.cores = 1, minSize = 7)

[1] "Applying the Zscore method"
Setting parallel calculations through a MulticoreParam back-end
with workers=1 and tasks=100.
Estimating combined z-scores for 3035 gene sets.
Setting parallel calculations through a MulticoreParam back-end
with workers=1 and tasks=100.
Estimating combined z-scores for 3035 gene sets.
Setting parallel calculations through a MulticoreParam back-end
with workers=1 and tasks=100.
Estimating combined z-scores for 3036 gene sets.
Setting parallel calculations through a MulticoreParam back-end
with workers=1 and tasks=100.
Estimating combined z-scores for 3037 gene sets.
Setting parallel calculations through a MulticoreParam back-end
with workers=1 and tasks=100.
Estimating combined z-scores for 3027 gene sets.
```

The result obtained is the proper object to perform meta-analysis (objectMApath). To

3 Performing Meta-analysis

GSEMA package has implemented gene set enrichment meta-analysis techniques based on the combination of effect sizes in the *metaAnalysisDEpath()* function. Although this function can be applied directly, it is advisable to consider some previous steps.

3.1 Filtering paths with low expression

Before performing the meta-analysis, it is important to filter out those pathways with low expression in the datasets. This is because the results of the meta-analysis could be biased by the presence of pathways with low expression, which may lead to an increase in the number of false positives. **GSEMA** provides a function called *filterPaths()* that allows to filter out those pathways with low expression. The inputs of this function are:

- *objectMApath*: the meta-analysis object of **GSEMA** package.
- *threshold*:A number that indicates the threshold to eliminate a gene set. For a eliminate a gene set is necessary that the mean for both groups are less than the threshold. If threshold = "sd" the threshold will be the standard deviation of the gene set. The default value is 0.85.

```
> objectMApathSim <- filteringPaths(objectMApathSim, threshold = 0.85)
```

3.1.1 Calculation of the effects sizes

GSEMA has implemented three different estimator for calculating the effect sizes in each study:

- The **"limma"** method used the *limma* package [6] to calculate the effect size and the variance of the effect size. The effect size is calculated from the moderated Student's t computed by *limma*. From it, the estimator of Hedges'g and its corresponding variance are obtained. In this way, some of the false positives obtained by the "SMD" method are reduced.
- The **"SMD"** (Standardized mean different) method calculates the effect size using the Hedges'g estimator [7].
- The **"MD"** (raw mean different) calculates the effects size as the difference between the means of the two groups [8].

3.2 Performing meta-analysis: *metaAnalysisDEpath()*

The *metaAnalysisDEpath()* function allows to perform a meta-analysis in only one step, needing only the meta-analysis object created previously.

This function has as input:

- objectMApath: The meta-analysis object of **GSEMA** package.
- effectS: A list of two elements. The first element is a dataframe with gene sets in rows and studies in columns. Each component of the dataframe is the effect of a gene set in a study. The second element of the list is also a dataframe with the same structure, but in this case each component of the dataframe represent the variance of the effect of a gene set in a study. This argument should be only used in the case that objectMApath argument is null.
- measure A character string that indicates the type of effect size to be calculated. The options are "limma", "SMD" and "MD". The default value is "limma". See details for more information.
- WithinVarCorrect A logical value that indicates if the within variance correction should be applied. The default value is TRUE. See details for more information (cite)

- typeMethod: a character that indicates the method to be performed:
  - "FEM": Fixed Effects model.
  - "REM": Random Effects model.
- missAllow: a number between 0 and 1 that indicates the maximum proportion of missing values allows in a sample. If the sample has more proportion of missing values, the sample will be eliminated. In the other case, the missing values will be imputed by using the K-NN algorithm included in *impute* package [9]. In case the *objectMA* has been previously imputed, this element is not necessary.
- proportionData: a number between 0 and 1 that indicates the minimum proportion of datasets in which a gene must be contained to be included.In case the *objectMA* has been previously imputed, this element is not necessary.

In the following example, we have applied a Random Effect model to the *GSEMA* object ("objectMApathSim") we have been working with so far. In addition we have applied the "limma" method to calculate the effect size. Finally, we have allowed a 0.3 proportion of missing values in a sample and a gene set must have been contained in all studies:

```
> results <- metaAnalysisESpath(objectMApath = objectMApathSim,
+     measure = "limma", typeMethod = "REM", missAllow = 0.3, proportionData = 1)
[1] "Performing Random Effects Model"
```

The output of this function is a dataframe with the results of the meta-analysis where rows are the genes and columns are the different variables provided by the meta-analysis:

```
> results[1,]
                                    Pathway    Com.ES     ES.var     Qval
BIOCARTA_FMLP_PATHWAY BIOCARTA_FMLP_PATHWAY -2.542876 0.01754429 4.893471
                            tau2      Zval Pval FDR propDataset
BIOCARTA_FMLP_PATHWAY 0.01601656 -19.19805    0   0           1
> results[nrow(results),]
                          Pathway   Com.ES   ES.var     Qval tau2     Zval Pval
Pathway_Invented Pathway_Invented 33.30989 1.107186 2.494647    0 31.65651    0
                 FDR propDataset
Pathway_Invented   0           1
```

3.3 Meta analysis results

The results are provided in a dataframe with the variables:

- Com.ES: combined effect of the gene set.
- ES.var: variance of the combined effect of the gene set.
- Qval: total variance of the gene set.
- tau2: between-study variance of the gene set.
- zval: combined effect value for a standard normal. I can be use in order to find out if the gene set is overexpressed (positive value) or underexpressed (negative value).
- Pval: P-value of the meta-analysis for the gene set.

- FDR: P-value adjusted of the meta-analysis for the gene set.
- propDataset: Proportion of the datasets in which the gene is included.

3.4 Visualization of the results: heatmap

Finally, we can represent in a heatmap the significant gene sets in order to observe how they are regulated in each of the studies. In *heatmapPaths()* function we have to include both the object that has been used in the meta-analysis, the result of it and the applied method. In addition, this package offers three different scaling approaches (*scaling*) in order to compare properly the gene set enrichment scores of the studies in the heatmap:

- "zscor": It calculates a z-score value for each gene, that is, the mean gene expression from each gene is subtracted from each gene expression value and then it is divided by the standard deviation.
- "swr": Scaling relative to reference dataset approach [10].
- "rscale": It uses the rescale function of the *scales* package to scale the gene expresion [11].
- "none": no scaling approach is applied.

Moreover, in *regulation* argument, we can choose if we want to represent the over regulated or under regulated gene sets:

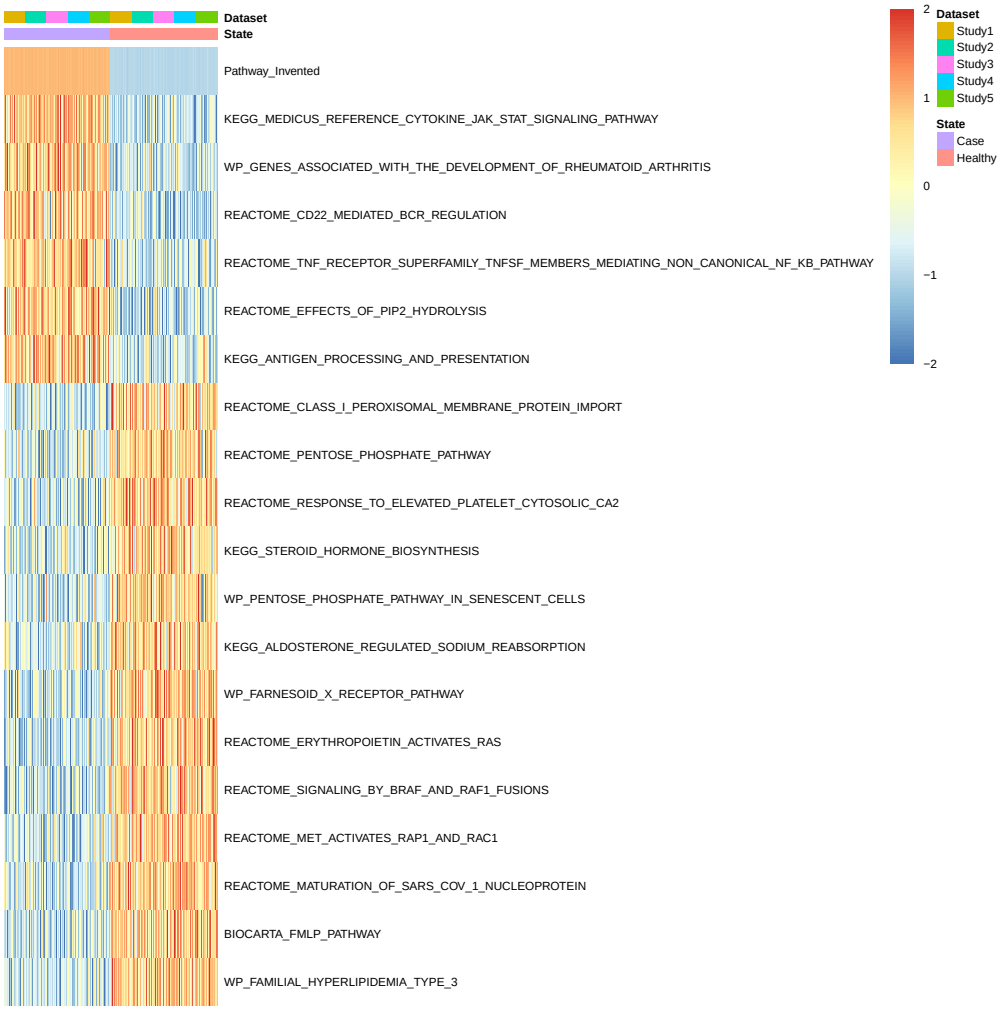
- "up": only up-regulated gene sets are represented.
- "down: only down-regulated gene sets are represented
- "all": up-regulated and down-regulated gene sets are represented.

We can choose the number of significant gene sets (*numSig*) that we want to be shown on the graph and the adjusted p-value from which a gene set is considered as significant (*fdrSig*). In addition, the gene sets that are not presented in one sample are represented in gray.

Here we present an example of the heatmap which have been obtained from the result of applying a random effects model to the object "*objectMApathSim*" and making use of a "zscor" scaling approach.

```
> res <- heatmapPaths(objectMA=objectMApathSim, results,
+             scaling = "zscor", regulation = "all",breaks=c(-2,2),
+             fdrSig = 0.05, comES_Sig = 1, numSig=20, fontsize = 5)

[1] "scaling using z-score..."
```



4 Additional information

4.1 Calculating individual Effects size

The *calculateESpath()* function returns the effects size in each of the studies. Moreover, it calculates the variance of each of the effects.

```
> Effects <- calculateESpath(objectMApath = objectMApathSim, measure = "limma")
> Effects[[1]][1,]

   Study1    Study2    Study3    Study4    Study5
-2.712938 -2.381185 -2.540136 -2.155484 -2.924636

> Effects[[2]][1,]

    Study1     Study2     Study3     Study4     Study5
0.07170491 0.07170491 0.07170491 0.07170491 0.07170491
```

5 Session info

```
R version 4.3.2 (2023-10-31)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.3 LTS

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=es_ES.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=es_ES.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=es_ES.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=es_ES.UTF-8 LC_IDENTIFICATION=C

time zone: Europe/Madrid
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] GSEMA_0.99.0

loaded via a namespace (and not attached):
  [1] mathjaxr_1.6-0              RColorBrewer_1.1-3
  [3] rstudioapi_0.15.0           magrittr_2.0.3
  [5] farver_2.1.1                rmarkdown_2.26
  [7] zlibbioc_1.48.2             vctrs_0.6.5
  [9] memoise_2.0.1               DelayedMatrixStats_1.24.0
 [11] RCurl_1.98-1.14             htmltools_0.5.8.1
 [13] S4Arrays_1.2.1              progress_1.2.3
 [15] Rhdf5lib_1.24.2             SparseArray_1.2.4
 [17] rhdf5_2.46.1                KernSmooth_2.23-22
 [19] DExMA_1.11.8                plyr_1.8.9
 [21] impute_1.76.0               cachem_1.0.8
 [23] igraph_2.0.3                lifecycle_1.0.4
 [25] iterators_1.0.14            pkgconfig_2.0.3
 [27] rsvd_1.0.5                  Matrix_1.6-3
 [29] R6_2.5.1                    fastmap_1.1.1
 [31] GenomeInfoDbData_1.2.11     MatrixGenerics_1.14.0
 [33] digest_0.6.35               numDeriv_2016.8-1.1
 [35] colorspace_2.1-0            AnnotationDbi_1.64.1
 [37] S4Vectors_0.40.2            singscore_1.22.0
 [39] swamp_1.5.1                 irlba_2.3.5.1
 [41] GenomicRanges_1.54.1        RSQLite_2.3.6
 [43] beachmat_2.18.1             metadat_1.2-0
```

```
 [45] fansi_1.0.6                 httr_1.4.7
 [47] abind_1.4-5                 mgcv_1.9-1
 [49] compiler_4.3.2              bit64_4.0.5
 [51] doParallel_1.0.17           metafor_4.6-0
 [53] BiocParallel_1.36.0         DBI_1.2.2
 [55] gplots_3.1.3.1              HDF5Array_1.30.1
 [57] MASS_7.3-60                 DelayedArray_0.28.0
 [59] caTools_1.18.2              gtools_3.9.5
 [61] tools_4.3.2                 glue_1.7.0
 [63] nlme_3.1-163                rhdf5filters_1.14.1
 [65] grid_4.3.2                  reshape2_1.4.4
 [67] generics_0.1.3              sva_3.50.0
 [69] gtable_0.3.4                tzdb_0.4.0
 [71] tidyr_1.3.1                 data.table_1.15.4
 [73] hms_1.1.3                   xml2_1.3.6
 [75] BiocSingular_1.18.0         ScaledMatrix_1.10.0
 [77] utf8_1.2.4                  XVector_0.42.0
 [79] BiocGenerics_0.48.1         foreach_1.5.2
 [81] pillar_1.9.0                stringr_1.5.1
 [83] GSVA_1.50.1                 limma_3.58.1
 [85] genefilter_1.84.0           splines_4.3.2
 [87] dplyr_1.1.4                 bnstruct_1.0.15
 [89] lattice_0.22-5              survival_3.5-7
 [91] bit_4.0.5                   GEOquery_2.70.0
 [93] annotate_1.80.0             tidyselect_1.2.1
 [95] SingleCellExperiment_1.24.0 locfit_1.5-9.9
 [97] Biostrings_2.70.3           pbapply_1.7-2
 [99] amap_0.8-19                 knitr_1.46
[101] IRanges_2.36.0              edgeR_4.0.16
[103] SummarizedExperiment_1.32.0 snpStats_1.52.0
[105] stats4_4.3.2                xfun_0.43
[107] Biobase_2.62.0              statmod_1.5.0
[109] DExMAdata_1.10.0            matrixStats_1.2.0
[111] pheatmap_1.0.12             stringi_1.8.3
[113] yaml_2.3.8                  evaluate_0.23
[115] codetools_0.2-19            tibble_3.2.1
[117] BiocManager_1.30.22         graph_1.80.0
[119] cli_3.6.2                   xtable_1.8-4
[121] munsell_0.5.1               Rcpp_1.0.12
[123] GenomeInfoDb_1.38.7         png_0.1-8
[125] XML_3.99-0.16.1             parallel_4.3.2
[127] readr_2.1.5                 ggplot2_3.5.0
[129] blob_1.2.4                  prettyunits_1.2.0
[131] sparseMatrixStats_1.14.0    bitops_1.0-7
[133] GSEABase_1.64.0             scales_1.3.0
[135] purrr_1.0.2                 crayon_1.5.2
[137] BiocStyle_2.30.0            rlang_1.1.3
[139] KEGGREST_1.42.0
```

References

[1] Toro-Domínguez D., Villatoro-García J.A., Martorell-Marugán J., and et al. A survey of gene expression meta-analysis:methods and applications. *Briefings in Bioinformatics*, pages 1–12, 2020. doi:https://doi.org/10.1093/bib/bbaa019.

[2] Hänzelmann S and Castelo Ra nd Guinney J. Gsva: gene set variation analysis for microarray and rna-seq data. *BMC Bioinformatics*, page 7, 2013. doi:https://doi.org/10.1186/1471-2105-14-7.

[3] Lee E, Chuang HY, Kim JW, Ideker T, and Lee D. Inferring pathway activity toward precise disease classification. *PLOS Computational Biology*, page e1000217, 2008. doi:https://doi.org/10.1371/journal.pcbi.1000217.

[4] Barbie DA, Tamayo P, Boehm JS, Kim SY, Moody SE, Dunn IF, and et al. Systematic rna interference reveals that oncogenic kras-driven cancers require tbk1. *Nature*, pages 108–112, 2009. doi:https://doi.org/10.1371/journal.pcbi.1000217.

[5] Foroutan M, Bhuva DD, Lyu R, Horan K, Cursons J, and Davis MJ. Single sample scoring of molecular phenotypes. *BMC Bioinformatics*, page 404, 2018. doi:https://doi.org/10.1186/s12859-018-2435-4.

[6] Smyth G. K., Ritchie M., Thorne N., Yifang Hu, and et al. Linear models for microarray and rna-seq data user's guide. 2020.

[7] Hedges LV. Distribution theory for glass's estimator of effect size and related estimators. *Journal of Educational Statistics*, pages 107–128, 1981. doi:https://doi.org/10.2307/1164588.

[8] Borenstein M, Hedges LV, Higgins JPT, and Rothstein. Introduction to meta-analysis. *John Wiley and Sons*, 2021.

[9] Hastie T., Tibshirani R., Narasimhan B., and Chu G. impute: Imputation for microarray data. 2019.

[10] Lazar C., Meganck S., Taminau J., and et al. Batch effect removal methods for microarray gene expression data integration: a survey. *Briefings in Bioinformatics*, pages 469–490, 2013. doi:10.1093/bib/bbs037.

[11] Wickham H. and Siedel D. Scale functions for visualization. 2020.