iNETgrate: Integrating gene expression and DNA methylation data in a gene network

Isha Mehta and Habil Zare

Modified: 5 December, 2021. Compiled: May 1, 2024

Contents

1 Introduction

The biological processes that take place within a cell often require intricate coordination among *multiple* genes and proteins [1]. Thus, the traditional approaches that study associations between individual genes and a phenotype cannot provide a full understanding of these complex biological phenomena [2, 3]. These approaches fail to pinpoint the biological mechanisms of complicated conditions such as cancer because complex diseases are usually caused by collaboration of more than a few genes. In particular, the detection of the subtle but coordinated and consistent changes in the expression levels of a set of functionally related genes is often more important and informative than detecting dramatic changes in the expression levels of a few individual genes [1].

Network analysis can detect subtle but consistent changes in a set of interacting and functionally related genes [1]. The more data used to build the network, the better the network obtained because the resulting network models the interaction between genes more accurately. However, integrating different types of omics data into a single network can be challenging. For example, DNA methylation is usually measured on hundreds of thousands of loci, and there is no one–to–one correspondence between these loci and genes, which are the nodes of the coexpression network. The *iNETgrate* package is specifically developed to fuse (i.e., effectively integrate) DNA methylation data with gene expression data in a single network [4]. The nodes (vertices) of the network are genes and the connection (edge) between two nodes is weighted based on both gene expression and DNA methylation data. The *iNETgrate* package is a significant enhancement over our *Pigengene* approach [5] because it effectively integrates DNA methylation data into the gene network [4].

A practical exemplar application of *iNETgrate* is in prognostication of acute myeloid leukemia (AML), which is a cancer of the blood and bone marrow [4]. Most AML patients are classified as low–, intermediate–, and high–risk. There are urgent and effective treatments for high– and low–risk patients, such as bone marrow transplant and chemotherapy, respectively. The intermediate–risk group is a mixture of high– and low–risk patients but, their actual risk level is not detectable based on current methods used in clinics. Since the survival rate of these patients cannot be predicted precisely, it is not possible to choose the most efficient treatment for them. Therefore, there is a demand to find a way to reclassify these patients into either high– or low–risk groups based on their clinical and molecular omics data. In most methods, gene expression is the primary data source to find risk groups. Some research groups used mutation and molecular abnormalities along with gene expression and found risk groups of more patients. Here, we illustrate using gene expression and DNA methylation to reclassify intermediate–risk patients.

2 How to run *iNETgrate* ?

2.1 Installation

iNETgrate is an *R* package that can be downloaded and installed from *Bioconductor* by using following commands in R:

```
if (!requireNamespace("BiocManager", quietly=TRUE)) install.packages("BiocManager")
BiocManager::install("iNETgrate")
```

Alternatively, if the built package is already available, it can be installed by the following command in Unix:

```
R CMD INSTALL iNETgrate_x.y.z.tar.gz
```

where x.y.z determines the version. The second approach requires all the dependencies be installed manually, therefore, the first approach is preferred.

2.2 A quick overview

iNETgrate constructs a weighted gene network where the weights of the edges is calculated by a combined correlation scores of gene expression and DNA methylation beta values, i.e.,

$$\mathcal{W}(g_i, g_j) = (1-\mu)\big(\big|cor_E(g_i, g_j)\big|\big) + \mu\big(\big|cor_M(g_i, g_j)\big|\big), \qquad (1)$$

where g_i and g_j is a pair of genes for which edge weight is to be calculated, μ is a value between 0 and 1 given by the user, $\big|cor_E(g_i, g_j)\big|$ is the absolute correlation between expression of the gene pair, and similarly, $\big|cor_M(g_i, g_j)\big|$ is the absolute correlation between beta values of the gene pair.

iNETgrate then identifies gene modules (i.e., clusters of coexpressed and comethylated genes), computes an eigengene for each module, and uses these biological signatures as features for classifying patients into risk categories. The main function is `iNETgrate` which requires a gene expression profile, a DNA methylation beta value profile and the corresponding conditions (types). Individual functions are also provided to facilitate running the pipeline in a customized way. The inferred biological signatures (eigengenes) are useful for further supervised or unsupervised analyses.

2.3 What is an eigengene?

In most functions of this package, eigenegenes are computed or used as robust biological signatures. Briefly, each eigengene $\mathcal{E}$ is a weighted average of the expression of all genes in a given set of n genes (also known as a gene module or a cluster of genes).

$$\mathcal{E} = \alpha_1 g_1 + \alpha_2 g_2 + \cdots + \alpha_i g_i \tag{2}$$

where α_i represents the weight corresponding to gene g_i. The weights are adjusted using PCA in a way that the explained variance is maximized. This guarantees that the loss in the biological information in minimized.

2.4 What is an eigenloci?

To compute an effective DNA methylation value at gene level, we calculate a weighted average of beta values per gene. Briefly, each eigenloci is a weighted average of the beta values of the probes (loci) corresponding to the gene of interest. If the number of loci for a gene is less than six, all of them contribute to the eigenloci with some weight, which is determined automatically. Otherwise, we identify and use a subset of highly correlating loci (i.e., "core") to compute the eigenloci. A gene expression profile and an eigenloci profile are the two key inputs to the `makeNetwork` function.

2.5 A toy example

For a quick start, we demonstrate the application of *iNETgrate* pipeline on a leukemia dataset below [4]. The first step is to load the package and data in *R*:

```
library(iNETgrate)

## Loading required package:  BiocStyle

##

##

## Registered S3 method overwritten by 'gplots':
##   method         from
##   reorder.factor gdata

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)

set.seed(1)
```

In this vignette, we use the toy data that is included in the package. Please note that the provided data in the package is sub–sampled for a quick demonstration. For real applications, the expression of thousands of genes should be used. Analyzing such input with *iNETgrate* can take a few hours and may require 5-10 GB of memory to save the results.

The following commands run *iNETgrate* pipeline on the toy data. First, we load the toy data.

```
data(toyRawAml)
class(toyRawAml)

## [1] "list"

names(toyRawAml)

## [1] "genExpr"           "genExprSampleInfo" "rawDnam"           "clinical"
```

Then, we define the clinical settings and run *iNETgrate* .

```
clinSettings <- c("patientIDCol"="bcr_patient_barcode",
"eventCol"="vital_status",
"timeCol"="days_to_last_followup",
"riskCatCol"="acute_myeloid_leukemia_calgb_cytogenetics_risk_category",
"riskFactorCol"="cytogenetic_abnormalities",
"event"="Dead",
"riskHigh"="Poor",
"riskLow"="Favorable")
print(clinSettings)

##                                              patientIDCol
##                                     "bcr_patient_barcode"
##                                                  eventCol
##                                            "vital_status"
##                                                   timeCol
##                                   "days_to_last_followup"
##                                                riskCatCol
## "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
##                                             riskFactorCol
##                               "cytogenetic_abnormalities"
##                                                     event
##                                                    "Dead"
##                                                  riskHigh
##                                                    "Poor"
##                                                   riskLow
##                                                "Favorable"
```

See the documentation of the `iNETgrate` function for a toy example like the following:

```
inetgrator <- iNETgrate(Data=toyRawAml, clinSettings=clinSettings,
                        saveDir="iNETgrateOut", mus=0.6)
```

Results and figures are saved in the "iNETgrateOut" directory under the current directory. For more advanced applications, the user is encouraged to analyze the data step-by-step and customize the individual functions such as `makeNetwork` and `computEigengenes`.

2.6 Running the *iNETgrate* pipeline step by step

If you are curious about the specific steps in the *iNETgrate* pipeline, or you need to run some steps with different settings, you can follow the steps below. With the default values, the results will be similar to the output of the `iNETgrate` function.

2.6.1 Setting paths

Before starting the analysis, we need to create some directory to save the plots and results.

```
resPath <- file.path(tempdir(), "iNETgrateRes") ## the result path
message(paste("Results will be saved in:", resPath, sep="\n"))

## Results will be saved in:
## /tmp/RtmpWJKGUw/iNETgrateRes

netPath <- file.path(resPath, "net")
survivalPath <- file.path(netPath, "surv")
dir.create(survivalPath, showWarnings=FALSE, recursive=TRUE)
```

2.6.2 Cleaning data

The first step of the *iNETgrate* pipeline is to clean the input data using the `cleanAllData` function. Here we make sure that the gene expression and DNA methylation matrices have row and column names, and do not include too many `NA` values. We impute missing values in the DNA methylation data if needed. Also, we clean and subset clinical data.

```
riskCatCol <- "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
riskFactorCol <- "cytogenetic_abnormalities"
cleanedToy <- cleanAllData(genExpr=toyRawAml$genExpr,
                            genExprSampleInfo=toyRawAml$genExprSampleInfo,
                            rawDnam=toyRawAml$rawDnam, savePath=resPath,
                            clinical=toyRawAml$clinical,
                            riskCatCol=riskCatCol, riskFactorCol=riskFactorCol,
                            riskHigh="Poor", riskLow="Favorable",
                            verbose=1)
```

The output must be saved by the user if they intent to use the cleaned data later. The output is a list of `4` components namely `genExpr`, `dnam`, `locus2gene` and `survival`. `survival` is a subset of clinical data, where rows are patients and columns are vitality and survival related information. `genExpr` is a matrix of gene expression profile, where rows are genes and columns are patient IDs with sample type attached in cases where multiple sample types exists. `dnam` is a matrix of DNA methylation profile, where rows are loci and columns are patient IDs with sample type attached in cases where multiple sample types exists. DNA methylation data is processed using `preprocess.Dnam` for missing beta values, and removing some non-CpG loci or SNP enriched loci.Finally `locus2gene` is a dataframe where rows are loci and columns provide information on the related gene.

This is followed by filtering and omitting genes or loci that have too low correlation with survival data and vital status. The `filterLowCor` function computes these correlations and selects the desired data. We then find a combined set of genes that have some correlation with survival time or vital status based on expression or DNA methylation profiles using `computeUnion`. The above–mentioned filtering and combining steps can be performed using the function `electGenes`, which is a wrapper for `filterLowCor` and `computeUnion` functions.

```
data(toyCleanedAml)
cleaned <- toyCleanedAml
elected <- electGenes(genExpr=cleaned$genExpr, dnam=cleaned$dnam,
                       survival=cleaned$survival, savePath=resPath,
                       event="Dead", locus2gene=cleaned$locus2gene,
                       doAllLoci=FALSE, verbose=1)

## Filtering gene expression data...

## Cleaning all data started at:  2024-05-01 17:48:49 EDT

## Both minCor and ratio are passed as inputs, both thresholds will
be used.
```

```
## 3 entities with NA correlation to survival
##  time are removed.

## 78 entities in expression are selected
##  as their absolute correlation with survival time > 0.2

## 12  cases are
##  removed because of NA in vital status

## 1 entities with NA correlation to vital status
##  are removed.

## 28 entities in expression are
##  selected as their absolute correlation with vital status > 0.2

## Number of selected data in expression based on
##  correlation threshold 0.2 for vital status AND
##  survival time:  102 .

## Keeping only 34 entities in expression because ratio is set to 0.333333333333333
.

## The correlation plots are saved at:
##  /tmp/RtmpWJKGUw/iNETgrateRes/plots

## Filtering DNA methylation data...

## Cleaning all data started at:  2024-05-01 17:48:49 EDT

## 0 entities with NA correlation to survival
##  time are removed.

## 55 entities in dnam are selected
##  as their absolute correlation with survival time > 0.2

## 14  cases are
##  removed because of NA in vital status

## 0 entities with NA correlation to vital status
##  are removed.

## 16 entities in dnam are
##  selected as their absolute correlation with vital status > 0.2

## Number of selected data in dnam based on
##  correlation threshold 0.2 for vital status AND
##  survival time:  71 .

## Keeping only 71 entities in dnam because ratio is set to 1 .
## The correlation plots are saved at:
##  /tmp/RtmpWJKGUw/iNETgrateRes/plots

## Combining filtered data...

## unionGenes set size:  77
```

The output of `electGenes` is a list of `5` components including `unionGenes`, which is a character vector of gene IDs that have some correlation with survival time or vital status based on expression or DNA methylation profiles. This union gene set is further used to makeNetwork and compute eigengenes in next steps.

2.6.3 Computing DNA methylation values at the gene level

Computing effective DNA methylation profile at gene level is necessary to construct a combined network where the nodes of the network are genes. Hence, we compute eigenloci described earlier.

```
patientLabel <- setNames(as.character(cleaned$survival$Risk1),
                         nm=rownames(cleaned$survival))
inBoth <- intersect(colnames(cleaned$dnam), names(patientLabel))
computedEloci <- computEigenloci(dnam=cleaned$dnam[ ,inBoth],
    Labels=patientLabel[inBoth],
    geNames=elected$unionGenes,
    locus2gene=cleaned$locus2gene,
    plotPath=resPath,
    Label1="Low", Label2="High",
    verbose=1)

## Computing gene2locus...

## For every one of the 77 genes...
```

2.6.4 Making the combined network

Now, our data is ready to make a network using `makeNetwork`. Here, we use a `mu` value to combine adjacency matrices from the expression and DNA methylation (i.e., `eigenloci`) datasets into a single network of genes (Eq. 1). In real applications, several values for `mu` can be used in a numeric vector and then the best value will be determined in later steps of the analyzeSurvival.

```
eigenloci <- computedEloci$eigenloci
madeNet <- makeNetwork(genExpr=cleaned$genExpr, eigenloci=eigenloci,
                            geNames=elected$unionGenes, mus=0.6,
                            doRemoveTOM=TRUE, outPath=netPath,
                            verbose=1)

## Warning:  executing %dopar% sequentially:  no parallel backend registered

print(madeNet$mu2modules[,1:5, drop=FALSE])

##     BAI1 CAPNS1 CHD1 CYC1 EXT1
## 0.6    0      2    1    0    0
```

The modules for each value of the input `mu` are identified and returned in `madeNet$mu2modules`, which is a matrix with the `mu` values on rows and genes on columns. The entries of the matrix are integer values determining the module number a gene belongs to.

2.6.5 Computing eigengenes

The next step is to compute a representative value (feature) for each identified module (i.e., an eigengene) based on gene expression and optionally, DNA methylation data.

```
e1 <- computEigengenes(genExpr=cleaned$genExpr, eigenloci=eigenloci,
                          netPath=netPath, geNames=elected$unionGenes,
                          Labels=patientLabel, Label1="Low",
                          Label2="High", mus=c(0.6), combiningMu=NA,
                          doIgnoreNas=TRUE, survival=cleaned$survival,
                          event="Dead", verbose=1,
                          mu2modules=madeNet$mu2modules)

## mu value:  0.6  lCombine:  1

## Using expr

## Add dnam

## Projecting...

## mu value:  0.6  lCombine:  0

## Using expr

## Add dnam

## Projecting...

## mu value:  0.6  lCombine:  0.6

## Using expr
## Add dnam

## Projecting...
```

2.6.6 Survival analysis

Finally, our Cox regression analysis identifies the module (gene cluster) or the combination of up to 3 modules that are together most useful for prognostication. The identified modules are used in an accelerated failure time (AFT) model to classify the patients into different risk categories.

```
s1 <- analyzeSurvival(survival=cleaned$survival,
                       favRisk="High",
                       subSet="Int", mus=0.6,
                       netPath=netPath, outPath=survivalPath,
                       xmax1=15, xmin1=0, verbose=1)

## Accelerated failure time and Cox analysis ...

## inputEvent:

## inputEvent
##   0   1
##  65 108

## mu value :  0.6

## Loading objects:
##   eigengenes

##   [[ suppressing 10 column names 's0', 's1', 's2' ...  ]]
```

The above survival plots are saved in `survivalPath`. Each plot compares the KM curves among risk categories based on *iNETgrate* analysis, or the input labels from the clinical data, as mentioned in plot titles. The first two plots include all cases and the last two plots include only a subset of cases as described in each plot title. The confusion matrix is useful to compare the input labels vs. the risk categorization done by *iNETgrate* (i.e., the numbers on the diagonal show the agreement while other numbers show the discrepancy).

2.6.7 Inferring eigengenes in another dataset

If you want to infer the selected eigengenes in an independent dataset for validation, the `bestInetgrator` can be used to extract the weights in an organized list, which then can be used by the `inferEigengenes` function.

```
inetgrator <- bestInetgrator(bestPvalues=s1$bestPvalues,
                  usefuLoci=computedEloci$usefuLoci,
                  lociPigen=computedEloci$lociPigen,
                  netPath=netPath)

## Loading objects:
##   pigengene
```

2.6.8 Pathway analysis

Pathwyay overrepresentation analysis of the modules selected by our Cox analysis can be done using the *Pigengene*::`get.enriched.pw` function. For each selected module, a folder will be created, which includes an Excel file listing the pathways that are overrepresented by the genes in the corresponding module.

```
selectedModules <- names(inetgrator$modules)
geneList <- list()
for(m1 in selectedModules)
    geneList[[m1]] <- names(inetgrator$modules[[m1]]$genes)
library(org.Hs.eg.db) ## Needed for human genes.

## Loading required package:  AnnotationDbi

got <- Pigengene::get.enriched.pw(geneList, idType="SYMBOL",
                 pathwayDb="KEGG", outPath=survivalPath)

## Loading required package:  org.Mm.eg.db

##

## 'select()' returned 1:1 mapping between keys and columns

## Reading KEGG annotation online:  "https://rest.kegg.jp/link/hsa/pathway"...

## Reading KEGG annotation online:  "https://rest.kegg.jp/list/pathway/hsa"...
```

2.7 Citation

```
citation("iNETgrate")
```

To cite package 'iNETgrate' in publications use:

Sogand Sajedi et al.(2023) 'iNETgrate': integrating DNA methylation and gene expression data in a single gene network, Sajedi et al., In process. URL: https://bmcmedgenomics.biomedcentral.com/articles/10.1186/s12920-017-0253-6.

A BibTeX entry for LaTeX users is

@Article, author = Sogand Sajedi and et al., title = iNETgrate: integrating DNA methylation and gene expression data in a single gene network, journal = TBD, year = 2023, volume = ?, number = ?, pages = ?, month = ?,

3 Session Information

The output of `sessionInfo` on the system that compiled this document is as follows:

```
toLatex(sessionInfo())
```

- R version 4.4.0 RC (2024-04-16 r86468), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_GB`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `America/New_York`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 22.04.4 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.20-bioc/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.67.0, Biobase 2.65.0, BiocGenerics 0.51.0, BiocStyle 2.33.0, Biostrings 2.73.0, GenomeInfoDb 1.41.0, GenomicRanges 1.57.0, IRanges 2.39.0, IlluminaHumanMethylation450kanno.ilmn12.hg19 0.6.1, MatrixGenerics 1.17.0, S4Vectors 0.43.0, SummarizedExperiment 1.35.0, XVector 0.45.0, bumphunter 1.47.0, foreach 1.5.2, iNETgrate 1.3.0, iterators 1.0.14, locfit 1.5-9.9, matrixStats 1.3.0, minfi 1.51.0, org.Hs.eg.db 3.19.1, org.Mm.eg.db 3.19.1

- Loaded via a namespace (and not attached): BiocFileCache 2.13.0, BiocIO 1.15.0, BiocManager 1.30.22, BiocParallel 1.39.0, C50 0.1.8, Cubist 0.4.2.1, DBI 1.2.2, DOSE 3.31.0, DelayedArray 0.31.0, DelayedMatrixStats 1.27.0, Formula 1.2-5, GEOquery 2.73.0, GO.db 3.19.1, GOSemSim 2.31.0, GenomeInfoDbData 1.2.12, GenomicAlignments 1.41.0, GenomicFeatures 1.57.0, HDF5Array 1.33.0, HDO.db 0.99.1, Hmisc 5.1-2, Homo.sapiens 1.3.1, KEGGREST 1.45.0, KernSmooth 2.23-22, MASS 7.3-60.2, Matrix 1.7-0, ModelMetrics 1.2.2.2, OrganismDbi 1.47.0, Pigengene 1.31.0, R6 2.5.1, RBGL 1.81.0, RColorBrewer 1.1-3, RCurl 1.98-1.14, RSQLite 2.3.6, Rcpp 1.0.12, RcppParallel 5.1.7, RcppZiggurat 0.1.6, ReactomePA 1.49.0, Rfast 2.1.0, Rgraphviz 2.49.0, Rhdf5lib 1.27.0, Rsamtools 2.21.0, S4Arrays 1.5.0, SparseArray 1.5.0, TxDb.Hsapiens.UCSC.hg19.knownGene 3.2.2, UCSC.utils 1.1.0, WGCNA 1.72-5, XML 3.99-0.16.1, abind 1.4-5, annotate 1.83.0, ape 5.8, aplot 0.2.2, askpass 1.2.0, backports 1.4.1, base64 2.0.1, base64enc 0.1-3, beanplot 1.3.1, biomaRt 2.61.0, bit 4.0.5, bit64 4.0.5, bitops 1.0-7, blob 1.2.4, bnlearn 4.9.3, caTools 1.18.2, cachem 1.0.8, caret 6.0-94, checkmate 2.3.1, class 7.3-22, cli 3.6.2, cluster 2.1.6, clusterProfiler 4.13.0, codetools 0.2-20, colorspace 2.1-0, compiler 4.4.0, cowplot 1.1.3, crayon 1.5.2, curl 5.2.1, data.table 1.15.4, dbplyr 2.5.0, digest 0.6.35, doParallel 1.0.17, doRNG 1.8.6, dplyr 1.1.4, dynamicTreeCut 1.63-1, e1071 1.7-14, enrichplot 1.25.0, evaluate 0.23, fansi 1.0.6, farver 2.1.1, fastcluster 1.2.6, fastmap 1.1.1, fastmatch 1.1-4, fgsea 1.31.0, filelock 1.0.3, foreign 0.8-86, fs 1.6.4, future 1.33.2, future.apply 1.11.2, gdata 3.0.0, genefilter 1.87.0, generics 0.1.3, ggforce 0.4.2, ggfun 0.1.4, ggplot2 3.5.1, ggplotify 0.1.2, ggraph 2.2.1, ggrepel 0.9.5, ggtree 3.13.0, glmnet 4.1-8, globals 0.16.3, glue 1.7.0, gower 1.0.1, gplots 3.1.3.1, graph 1.83.0, graphite 1.51.0, graphlayouts 1.1.1, grid 4.4.0, gridExtra 2.3, gridGraphics 0.5-1, gson 0.1.0, gtable 0.3.5, gtools 3.9.5, hardhat 1.3.1, highr 0.10, hms 1.1.3, htmlTable 2.4.2, htmltools 0.5.8.1, htmlwidgets 1.6.4, httr 1.4.7, httr2 1.0.1, igraph 2.0.3, illuminaio 0.47.0, impute 1.79.0, inum 1.0-5, ipred 0.9-14, jsonlite 1.8.8, knitr 1.46, lattice 0.22-6, lava 1.8.0, lazyeval 0.2.2, libcoin 1.0-10, lifecycle 1.0.4, limma 3.61.0, listenv 0.9.1, lubridate 1.9.3, magrittr 2.0.3, mclust 6.1.1, memoise 2.0.1, multtest 2.61.0, munsell 0.5.1, mvtnorm 1.2-4, nlme 3.1-164, nnet 7.3-19, nor1mix 1.3-3, openssl 2.1.2, openxlsx 4.2.5.2, pROC 1.18.5, parallelly 1.37.1, partykit 1.2-20, patchwork 1.2.0, pheatmap 1.0.12, pillar 1.9.0, pkgconfig 2.0.3, plyr 1.8.9, png 0.1-8, polyclip 1.10-6, preprocessCore 1.67.0, prettyunits 1.2.0, prodlim 2023.08.28, progress 1.2.3, proxy 0.4-27, purrr 1.0.2, quadprog 1.5-8, qvalue 2.37.0, rappdirs 0.3.3, reactome.db 1.88.0, readr 2.1.5, recipes 1.0.10, reshape 0.8.9, reshape2 1.4.4, restfulr 0.0.15, rhdf5 2.49.0, rhdf5filters 1.17.0, rjson 0.2.21, rlang 1.1.3, rmarkdown 2.26, rngtools 1.5.2, rpart 4.1.23, rstudioapi 0.16.0, rtracklayer 1.65.0, scales 1.3.0, scatterpie 0.2.2, scrime 1.3.5, shadowtext 0.1.3, shape 1.4.6.1, siggenes 1.79.0, sparseMatrixStats 1.17.0, splines 4.4.0, statmod 1.5.0, stringi 1.8.3, stringr 1.5.1, survival 3.6-4, tibble 3.2.1, tidygraph 1.3.1, tidyr 1.3.1, tidyselect 1.2.1, tidytree 0.4.6, timeDate 4032.109, timechange 0.3.0, tools 4.4.0, treeio 1.29.0, tweenr 2.0.3, txdbmaker 1.1.0, tzdb 0.4.0, utf8 1.2.4, vctrs 0.6.5, viridis 0.6.5, viridisLite 0.4.2, withr 3.0.0, xfun 0.43, xml2 1.3.6, xtable 1.8-4, yaml 2.3.8, yulab.utils 0.1.4, zip 2.3.1, zlibbioc 1.51.0

References

[1] Dong-Yeon Cho, Yoo-Ah Kim, and Teresa M Przytycka. Network biology approach to complex diseases. *PLoS Comput Biol*, 8(12):e1002820, 2012.

[2] Lewis G Halsey, Douglas Curran-Everett, Sarah L Vowler, and Gordon B Drummond. The fickle p value generates irreproducible results. *Nature methods*, 12(3):179–185, 2015.

[3] YounJeong Choi and Christina Kendziorski. Statistical methods for gene set coexpression analysis. *Bioinformatics*, 25(21):2780–2786, 2009.

[4] Hanie Samimi, Isha Mehta, Thomas Roderick Docking, Aamir Zainulabadeen, Aly Karsan, and Habil Zare. Dna methylation analysis improves the prognostication of acute myeloid leukemia. *eJHaem*, 2(2):211–218, 2021.

[5] A Foroushani, R Agrahari, R Docking, A Karsan, and H Zare. Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia. *In preparation*.

[6] Ken I Mills, Alexander Kohlmann, P Mickey Williams, Lothar Wieczorek, Wei-min Liu, Rachel Li, Wen Wei, David T Bowen, Helmut Loeffler, Jesus M Hernandez, et al. Microarray-based classifiers and prognosis models identify subgroups with distinct clinical outcomes and high risk of aml transformation of myelodysplastic syndrome. *Blood*, 114(5):1063–1072, 2009.

[7] Ka Yee Yeung and Walter L. Ruzzo. Principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[8] Bin Zhang, Chris Gaiteri, Liviu-Gabriel Bodea, Zhi Wang, Joshua McElwee, Alexei A Podtelezhnikov, Chunsheng Zhang, Tao Xie, Linh Tran, Radu Dobrin, et al. Integrated systems approach identifies genetic nodes and networks in late-onset alzheimer's disease. *Cell*, 153(3):707–720, 2013.