IFAA

IFAA offers a robust approach to make inference on the association of covariates with the absolute abundance (AA) of microbiome in an ecosystem. It can be also directly applied to relative abundance (RA) data to make inference on AA because the ratio of two RA is equal ratio of their AA. This algorithm can estimate and test the associations of interest while adjusting for potential confounders. High-dimensional covariates are handled with regularization. The estimates of this method have easy interpretation like a typical regression analysis. High-dimensional covariates are handled with regularization and it is implemented by parallel computing. False discovery rate is automatically controlled by this approach. Zeros do not need to be imputed by a positive value for the analysis. The IFAA package also offers the 'MZILN' function for estimating and testing associations of abundance ratios with covariates.

To model the association, the following equation is used:

$$\log(\mathcal{Y}_i^k)|\mathcal{Y}_i^k > 0 = \beta^{0k} + X_i^T\beta^k + W_i^T\gamma^k + Z_i^T b_i + \epsilon_i^k, \quad k = 1, ..., K+1,$$

where

- $\mathcal{Y}_i^k$ is the AA of taxa k in subject i in the entire ecosystem.
- X_i is the covariate matrix.
- W_i is the confounder matrix.
- Z_i is the design matrix for random effects.
- β^k is the regression coefficients that will be estimated and tested with the `IFAA()` function.

The challenge in microbiome analysis is that we can not oberve $\mathcal{Y}_i^k$. What is observed is its small proportion: $Y_i^k = C_i\mathcal{Y}_i^k$ where C_i is an unknown number between 0 and 1 that denote the observed proportion. The IFAA method successfuly addressed this challenge.

Package installation

To install, type the following command in R console:

```
install.packages("IFAA", repos = "http://cran.us.r-project.org")
```

The package could be also installed from GitHub using the following code:

```
require(devtools)
devtools::install_github("gitlzg/IFAA")
```

Input for IFAA() function

Most of the time, users just need to feed the first three inputs to the function: `experiment_dat`, `testCov` and `ctrlCov`. All other inputs can just take their default values. Below are all the inputs of the functions

- `experiment_dat`: A SummarizedExperiment object containing microbiome data and covarites (see example on how to create a SummarizedExperiment object). The microbiome data can be absolute abundance or relative abundance with each column per sample and each row per taxon/OTU/ASV (or any other unit). No imputation is needed for zero-valued data points. The covarites data contains covariates and confounders with each row per sample and each column per variable. The covarites data has to be numeric or binary.

- `testCov`: Covariates that are of primary interest for testing and estimating the associations. It corresponds to X_i in the equation. Default is `NULL` which means all covariates are `testCov`.
- `ctrlCov`: Potential confounders that will be adjusted in the model. It corresponds to W_i in the equation. Default is `NULL` which means all covariates except those in `testCov` are adjusted as confounders.
- `sampleIDname`: Name of the sample ID variable in the data. In the case that the data does not have an ID variable, this can be ignored. Default is NULL.
- `testMany`: This takes logical value `TRUE` or `FALSE`. If `TRUE`, the `testCov` will contain all the variables in `CovData` provided `testCov` is set to be `NULL`. The default value is `TRUE` which does not do anything if `testCov` is not `NULL`.
- `ctrlMany`: This takes logical value `TRUE` or `FALSE`. If `TRUE`, all variables except `testCov` are considered as control covariates provided `ctrlCov` is set to be `NULL`. The default value is `FALSE`.
- `nRef`: The number of randomly picked reference taxa used in phase 1. Default number is `40`.
- `nRefMaxForEsti`: The maximum number of final reference taxa used in phase 2. The default is `2`.
- `refTaxa`: A vector of taxa names. These are reference taxa specified by the user to be used in phase 1 if the user believe these taxa are indepenent of the covariates. If the number of reference taxa is less than 'nRef', the algorithm will randomly pick extra reference taxa to make up 'nRef'. The default is `NULL` since the algorithm will pick reference taxa randomly.
- `adjust_method`: The adjusting method used for p value adjustment. Default is "BY" for dependent FDR adjustment. It can take any adjustment method for p.adjust function in R.
- `fdrRate`: The false discovery rate for identifying taxa/OTU/ASV associated with `testCov`. Default is `0.15`.
- `paraJobs`: If `sequentialRun` is `FALSE`, this specifies the number of parallel jobs that will be registered to run the algorithm. If specified as `NULL`, it will automatically detect the cores to decide the number of parallel jobs. Default is `NULL`.
- `bootB`: Number of bootstrap samples for obtaining confidence interval of estimates in phase 2 for the high dimensional regression. The default is `500`.
- `standardize`: This takes a logical value `TRUE` or `FALSE`. If `TRUE`, the design matrix for X will be standardized in the analyses and the results. Default is `FALSE`.
- `sequentialRun`: This takes a logical value `TRUE` or `FALSE`. Default is `FALSE`. This argument could be useful for debug.
- `refReadsThresh`: The threshold of proportion of non-zero sequencing reads for choosing the reference taxon in phase 2. The default is `0.2` which means at least 20% non-zero sequencing reads.
- `taxDropThresh`: The threshold of number of non-zero sequencing reads for each taxon to be dropped from the analysis. The default is `0` which means taxon without any sequencing reads will be dropped from the analysis.
- `SDThresh`: The threshold of standard deviations of sequencing reads for been chosen as the reference taxon in phase 2. The default is `0.05` which means the standard deviation of sequencing reads should be at least `0.05` in order to be chosen as reference taxon.
- `SDquantilThresh`: The threshold of the quantile of standard deviation of sequencing reads, above which could be selected as reference taxon. The default is `0`.
- `balanceCut`: The threshold of the proportion of non-zero sequencing reads in each group of a binary variable for choosing the final reference taxa in phase 2. The default number is `0.2` which means at least 20% non-zero sequencing reads in each group are needed to be eligible for being chosen as a final reference taxon.

- `seed`: Random seed for reproducibility. Default is `1`. It can be set to be NULL to remove seeding.

Output for IFAA() function

A list containing 2 elements

- `full_results`: The main results for IFAA containing the estimation and testing results for all associations between all taxa and all test covariates in `testCov`. It is a dataframe with each row representing an association, and eight columns named as "taxon", "cov", "estimate", "SE.est", "CI.low", "CI.up", "adj.p.value", and "sig_ind". The columns correspond to taxon name, covariate name, association estimates, standard error estimates, lower bound and upper bound of the 95% confidence interval, adjusted p value, and the indicator showing whether the association is significant after multiple testing adjustment.
- `metadata`: The metadata is a list containing the following items: `covariatesData`: A dataset containing covariates and confounders used in the analyses. `final_ref_taxon`: The final 2 reference taxa used for analysis. `ref_taxon_count`: The counts of selection for the associations of all taxa with test covariates in Phase 1. `ref_taxon_est`: The average magnitude estimates for the associations of all taxa with test covariates in Phase 1. `totalTimeMins`: Total time used for the entire analysis. `seed`: The seed used for the analysis for reproducibility. `fdrRate`: FDR rate used for the analysis. `adjust_method`: Multiple testing adjust method used for the analysis.

Example

The example datasets `dataM` and `dataC` are included in this package. The input should be a SummarizedExperiment object, which could be constructed as below.

If you already have a SummarizedExperiment format data, you can ignore the data processing steps below.

```
library(IFAA)
suppressMessages(library(SummarizedExperiment))

## load the example microbiome data. This could be relative abundance or
## absolute abundance data. If you have a csv or tsv file for the microbiome data,
## you can use read.csv() function or read.table() function in R to read the
## data file into R.
data(dataM)
dim(dataM)
#> [1] 40 61
dataM[1:5, 1:8]
#>   id rawCount1 rawCount2 rawCount3 rawCount4 rawCount5 rawCount6 rawCount7
#> 1  1         4        49         2         0       360       222         4
#> 2  2         0        20        14         0        86       211         5
#> 3  3         3         0         3         7         0        57         0
#> 4  4         9        18         5        31        42        58         8
#> 5  5         0         2         1        19        15        67         6

## load the example covariates data. If you have a csv or tsv file for the
## covariates data, you can use read.csv() function or read.table() function
## in R to read the data file into R.
data(dataC)
dim(dataC)
#> [1] 40  4
dataC[1:3, ]
#>   id        v1        v2        v3
#> 1  1  58.06969 -49.90376 -15.30643
```

```
#> 2  2  25.96522 -68.58894 -23.10992
#> 3  3 193.71625 124.40186 119.56747
```

Both the microbiome data `dataM` and the covariates data `dataC` contain 40 samples (i.e., 40 rows).

- `dataM` contains 60 taxa with absolute abundances and these are gut microbiome.
- `dataC` contains 3 covariates.

```
## Merge the microbiome data and covariate data by id to avoid unmatching observations.
data_merged<-merge(dataM,dataC,by="id",all=FALSE)

## Seperate microbiome data and covariate data, drop id variable from microbiome data
dataM_sub<-data_merged[,colnames(dataM)[!colnames(dataM)%in%c("id")]]
dataC_sub<-data_merged[,colnames(dataC)]

## Create a SummarizedExperiment object
test_dat<-SummarizedExperiment(assays=list(MicrobData=t(dataM_sub)), colData=dataC_sub)
```

If you already have a SummarizedExperiment format data, you can ignore the above steps. Next we analyze the data to test the association between microbiome and the variable `"v1"` while adjusting for the variables (potential confounders) `"v2"` and `"v3"`.

```
results <- IFAA(experiment_dat = test_dat,
                testCov = c("v1"),
                ctrlCov = c("v2","v3"),
                sampleIDname = c("id"),
                fdrRate = 0.05)
#> Data dimensions (after removing missing data if any):
#> 40 samples
#> 60 taxa/OTU/ASV
#> 1 testCov variables in the analysis
#> These are the testCov variables:
#> v1
#> 2 ctrlCov variables in the analysis
#> These are the ctrlCov variables:
#> v2, v3
#> 0 binary covariates in the analysis
#> 25.71 percent of microbiome sequencing reads are zero
#> Start Phase 1 analysis
#> 6 parallel jobs are registered for analyzing 40 reference taxa in Phase 1
#> 33 percent of phase 1 analysis has been done and it took 0.83 minutes
#> 6 parallel jobs are registered for analyzing 20 reference taxa in Phase 1
#> 67 percent of phase 1 analysis has been done and it took 1.34 minutes
#> 6 parallel jobs are registered for analyzing 20 reference taxa in Phase 1
#> 100 percent of phase 1 analysis has been done and it took 1.85 minutes
#> Start Phase 2 parameter estimation
#> 50 percent of Phase 2 is done and it took 0.019 minutes
#> Entire Phase 2 parameter estimation done and took 0.04 minutes.
#> The entire analysis took 1.89 minutes
```

In this example, we are only interested in testing the associations with `"v1"` which is why `testCov=c("v1")`. The variables `"v2" and "v3"` are adjusted as potential confounders in the analyses. The final analysis results are saved in the list `full_result` and the significant results can be extracted as follows:

```
summary_res<-results$full_result
sig_results<-subset(summary_res,sig_ind==T)
```

```
sig_results
#> DataFrame with 3 rows and 8 columns
#>          taxon         cov  estimate     SE.est    CI.low     CI.up adj.p.value
#>    <character> <character> <numeric>  <numeric> <numeric> <numeric>   <numeric>
#> 1  rawCount18          v1 0.0248316 0.00523955 0.0145620 0.0351011 1.92686e-04
#> 2  rawCount36          v1 0.0285573 0.00564242 0.0174982 0.0396165 5.61399e-05
#> 3  rawCount41          v1 0.0300483 0.00515387 0.0199467 0.0401499 1.49157e-06
#>      sig_ind
#>    <numeric>
#> 1         1
#> 2         1
#> 3         1
```

The results found three taxa "rawCount18", "rawCount36", "rawCount41" associated with "v1" while adjusting for "v2" and "v3". The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to β^k in the model equation.

The interpretation is that

- Every unit increase in "v1" is associated with approximately 2.5% increase in the absolute abundance of "rawCount18", approximately 2.9% increase in the absolute abundance of "rawCount36", and approximately 3.0% increase in the absolute abundance of "rawCount41" in the entire gut ecosystem.

Reference

Li et al.(2021) IFAA: Robust association identification and Inference For Absolute Abundance in microbiome analyses. Journal of the American Statistical Association. 116(536):1595-1608

MZILN() function

The IFAA package can also implement the Multivariate Zero-Inflated Logistic Normal (MZILN) regression model for estimating and testing the association of abundance ratios with covariates. The `MZILN()` function estimates and tests the associations of user-specified abundance ratios with covariates. When the denominator taxon of the ratio is independent of the covariates, 'MZILN()' should generate similar results as 'IFAA()'. The regression model of 'MZILN()' can be expressed as follows:

$$\log\left(\frac{\mathcal{Y}_i^k}{\mathcal{Y}_i^{K+1}}\right)|\mathcal{Y}_i^k > 0, \mathcal{Y}_i^{K+1} > 0 = \alpha^{0k} + \mathcal{X}_i^T\alpha^k + \epsilon_i^k, \;\; k = 1, ..., K,$$

where

- $\mathcal{Y}_i^k$ is the AA of taxa k in subject i in the entire ecosystem.
- $\mathcal{Y}_i^{K+1}$ is the reference taxon (specified by user).
- $\mathcal{X}_i$ is the covariate matrix for all covariates including confounders.
- α^k is the regression coefficients that will be estimated and tested.

Input for MZILN() function

Most of the time, users just feed the first three inputs to the function: `experiment_dat`, `refTaxa` and `allCov`. All other inputs can just take their default values. All the inputs for 'MZILN()' are:

- `experiment_dat`: A SummarizedExperiment object containing microbiome data and covarites (see example on how to create a SummarizedExperiment object). The microbiome data can be absolute abundance or relative abundance with each column per sample and each row per taxon/OTU/ASV (or any other unit). No imputation is needed for zero-valued data points. The covarites data contains

covariates and confounders with each row per sample and each column per variable. The covarites data has to be numeric or binary.

- `refTaxa`: Denominator taxa names specified by the user for the targeted ratios. This could be a vector of names.
- `allCov`: All covariates of interest (including confounders) for estimating and testing their associations with the targeted ratios. Default is 'NULL' meaning that all covariates in covData are of interest.
- `sampleIDname`: Name of the sample ID variable in the data. In the case that the data does not have an ID variable, this can be ignored. Default is NULL.
- `adjust_method`: The adjusting method for p value adjustment. Default is "BY" for dependent FDR adjustment. It can take any adjustment method for p.adjust function in R.
- `fdrRate` The false discovery rate for identifying ratios associated with `allCov`. Default is `0.15`.
- `paraJobs`: If `sequentialRun` is `FALSE`, this specifies the number of parallel jobs that will be registered to run the algorithm. If specified as `NULL`, it will automatically detect the cores to decide the number of parallel jobs. Default is `NULL`.
- `bootB`: Number of bootstrap samples for obtaining confidence interval of estimates for the high dimensional regression. The default is `500`.
- `taxDropThresh`: The threshold of number of non-zero sequencing reads for each taxon to be dropped from the analysis. The default is `0` which means taxon without any sequencing reads will be dropped from the analysis.
- `standardize`: This takes a logical value `TRUE` or `FALSE`. If `TRUE`, the design matrix for X will be standardized in the analyses and the results. Default is `FALSE`.
- `sequentialRun`: This takes a logical value `TRUE` or `FALSE`. Default is `TRUE`. It can be set to be "FALSE" to increase speed if there are multiple taxa in the argument 'refTaxa'.
- `seed`: Random seed for reproducibility. Default is `1`. It can be set to be NULL to remove seeding.

Output for MZILN() function

A list with two elements:

- `full_results`: The main results for MZILN containing the estimation and testing results for all associations between all taxa ratios with refTaxan being the denominator and all covariates in `allCov`. It is a dataframe with each row representing an association, and ten columns named as "ref_tax", "taxon", "cov", "estimate", "SE.est", "CI.low", "CI.up", "adj.p.value", "unadj.p.value" and "sig_ind". The columns correspond to the denominator taxon, numerator taxon, covariate name, association estimates, standard error estimates, lower bound and upper bound of the 95% confidence interval, adjusted p value, and the indicator showing whether the association is significant after multiple testing adjustment.
- `metadata`: The metadata is a list containing total time used in minutes, random seed used, FDR rate, and multiple testing adjustment method used.

Examples

We use the same example data The example dataset as that for illustrating the MZILN function. `dataM` and `dataC` are included in this package.

If you have a SummarizedExperiment format data, you can ignore the data processing steps below.

```
## load the example microbiome data. This could be relative abundance or
## absolute abundance data. If you have a csv or tsv file for the microbiome data,
## you can use read.csv() function or read.table() function in R to read the
```

```
## data file into R.
data(dataM)
dim(dataM)
#> [1] 40 61
dataM[1:5, 1:8]
#>   id rawCount1 rawCount2 rawCount3 rawCount4 rawCount5 rawCount6 rawCount7
#> 1  1         4        49         2         0       360       222         4
#> 2  2         0        20        14         0        86       211         5
#> 3  3         3         0         3         7         0        57         0
#> 4  4         9        18         5        31        42        58         8
#> 5  5         0         2         1        19        15        67         6

## load the example covariates data. If you have a csv or tsv file for the
## covariates data, you can use read.csv() function or read.table() function
## in R to read the data file into R.
data(dataC)
dim(dataC)
#> [1] 40  4
dataC[1:3, ]
#>   id        v1        v2        v3
#> 1  1  58.06969 -49.90376 -15.30643
#> 2  2  25.96522 -68.58894 -23.10992
#> 3  3 193.71625 124.40186 119.56747
```

Both the microbiome data `dataM` and the covariates data `dataC` contain 40 samples (i.e., 40 rows).

- `dataM` contains 60 taxa with absolute abundances and these are gut microbiome.
- `dataC` contains 3 covariates.

```
## load the example microbiome data. This could be relative abundance or
## absolute abundance data. If you have a csv or tsv file for the microbiome data,
## you can use read.csv() function or read.table() function in R to read the
## data file into R.
data_merged<-merge(dataM,dataC,by="id",all=FALSE)

## load the covariates data. If you have a csv or tsv file for the covariates data,
## you can use read.csv() function or read.table() function in R to read
## the data file into R.
dataM_sub<-data_merged[,colnames(dataM)[!colnames(dataM)%in%c("id")]]
dataC_sub<-data_merged[,colnames(dataC)]

## Create a SummarizedExperiment object
test_dat<-SummarizedExperiment(assays=list(MicrobData=t(dataM_sub)), colData=dataC_sub)
```

If you already have a SummarizedExperiment format data, you can ignore the above steps. Next we analyze the data to test the associations between the ratio "rawCount18/rawCount11" and all the three variables `"v1"`, `"v2"` and `"v3"` in a multivariate model where all `"v1"`, `"v2"` and `"v3"` are independent variables simultaneously.

```
results <- MZILN(experiment_dat=test_dat,
                 refTaxa=c("rawCount11"),
                 allCov=c("v1","v2","v3"),
                 sampleIDname = c("id"),
                 fdrRate=0.05)
#> Data dimensions (after removing missing data if any):
```

```
#> 40 samples
#> 60 taxa/OTU/ASV
#> 3 testCov variables in the analysis
#> These are the testCov variables:
#> v1, v2, v3
#> 0 ctrlCov variables in the analysis
#> 0 binary covariates in the analysis
#> 25.71 percent of microbiome sequencing reads are zero
#> Estimation done for the 1th denominator taxon: rawCount11 and it took 0.02 minutes
#> The entire analysis took 0.02 minutes
```

The full final analysis results can be extracted as follows:

```
summary_res<-results$full_results
```

The results for the log-ratio of "rawCount18" over "rawCount11" can extracted as follows:

```
summary_res[summary_res$taxon=="rawCount18",,drop=FALSE]
#> DataFrame with 3 rows and 10 columns
#>       ref_tax       taxon         cov    estimate     SE.est      CI.low
#>   <character> <character> <character>   <numeric>  <numeric>   <numeric>
#> 1  rawCount11  rawCount18          v1  0.02310369 0.00557079  0.01218495
#> 2  rawCount11  rawCount18          v2  0.00260412 0.00317369 -0.00361632
#> 3  rawCount11  rawCount18          v3 -0.00625053 0.00281432 -0.01176659
#>          CI.up adj.p.value unadj.p.value   sig_ind
#>      <numeric>   <numeric>     <numeric> <logical>
#> 1  0.034022440  0.00308539   0.000033643      TRUE
#> 2  0.008824558  1.00000000   0.411913110     FALSE
#> 3 -0.000734468  0.80559644   0.026352611     FALSE
```

The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to α^k in the model equation, and can be interpreted as the associations between the covariates and log-ratio of `"rawCount18"` over ' "rawCount11" '.

The interpretation for the results is that

- Every unit increase in `"v1"` is associated with approximately 2.3% increase in the abundance ratio of `"rawCount18"` over `"rawCount11"` (while controlling for `"v2"` and `"v3"`); Every unit increase in `"v2"` is associated with approximately 0.26% increase in the abundance ratio of `"rawCount18"` over `"rawCount11"` (while controlling for `"v1"` and `"v3"`), but not statistically significant; Every unit increase in `"v3"` is associated with approximately -0.63% decrease in the abundance ratio of `"rawCount18"` over `"rawCount11"` (while controlling for `"v1"` and `"v2"`), but not statistically significant.

We can also extract all the ratios (with `"rawCount11"` being the denominator taxon) that are significantly associated with any of the covariates as follows:

```
subset(summary_res,sig_ind==T)
#> DataFrame with 3 rows and 10 columns
#>       ref_tax       taxon         cov  estimate     SE.est    CI.low     CI.up
#>   <character> <character> <character> <numeric>  <numeric> <numeric> <numeric>
#> 1  rawCount11  rawCount18          v1 0.0231037 0.00557079 0.0121849 0.0340224
#> 2  rawCount11  rawCount36          v1 0.0296571 0.00589475 0.0181034 0.0412108
#> 3  rawCount11  rawCount41          v1 0.0256273 0.00546293 0.0149199 0.0363346
#>   adj.p.value unadj.p.value   sig_ind
#>     <numeric>     <numeric> <logical>
#> 1 0.003085391   3.36430e-05      TRUE
#> 2 0.000134166   4.87647e-07      TRUE
```

```
#> 3 0.000373778   2.71711e-06       TRUE
```

The interpretation for the results is that

- Every unit increase in "v1" is associated with approximately 2.3% increase in the abundance ratio of "rawCount18" over "rawCount11" (while controlling for "v2" and "v3"), and it is statistically significant; Every unit increase in "v1" is also associated with approximately 3.0% increase in the abundance ratio of "rawCount36" over "rawCount11" (while controlling for "v2" and "v3"), and it is statistically significant; Every unit increase in "v1" is also associated with approximately 2.6% decrease in the abundance ratio of "rawCount41" over "rawCount11" (while controlling for "v2" and "v3"), and it is statistically significant.

Reference

Li et al.(2018) Conditional Regression Based on a Multivariate Zero-Inflated Logistic-Normal Model for Microbiome Relative Abundance Data. Statistics in Biosciences 10(3): 587-608

Session Info

```
sessionInfo()
#> R version 4.2.1 (2022-06-23 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 19044)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=C
#> [2] LC_CTYPE=English_United States.utf8
#> [3] LC_MONETARY=English_United States.utf8
#> [4] LC_NUMERIC=C
#> [5] LC_TIME=English_United States.utf8
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] SummarizedExperiment_1.26.1 Biobase_2.56.0
#>  [3] GenomicRanges_1.48.0        GenomeInfoDb_1.32.2
#>  [5] IRanges_2.30.0              S4Vectors_0.34.0
#>  [7] BiocGenerics_0.42.0         MatrixGenerics_1.8.0
#>  [9] matrixStats_0.62.0          IFAA_1.0.7
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_1.0.7             mvtnorm_1.1-3          lattice_0.20-45
#>  [4] class_7.3-20           glmnet_4.1-3           digest_0.6.29
#>  [7] RhpcBLASctl_0.21-247.1 foreach_1.5.1          parallelly_1.30.0
#> [10] slam_0.1-49            R6_2.5.1               cellranger_1.1.0
#> [13] sparsesvd_0.2          qlcMatrix_0.9.7        evaluate_0.14
#> [16] rootSolve_1.8.2.3      e1071_1.7-11           httr_1.4.2
#> [19] zlibbioc_1.42.0        rlang_1.0.3            Exact_3.1
#> [22] readxl_1.4.0           rstudioapi_0.13        data.table_1.14.2
#> [25] Matrix_1.4-1           rmarkdown_2.12         mathjaxr_1.4-0
#> [28] splines_4.2.1          stringr_1.4.0          RCurl_1.98-1.7
```

```
#> [31] DelayedArray_0.22.0    proxy_0.4-27           compiler_4.2.1
#> [34] xfun_0.29              speedglm_0.3-4         shape_1.4.6
#> [37] DescTools_0.99.45      htmltools_0.5.2        lmom_2.9
#> [40] GenomeInfoDbData_1.2.8 MatrixExtra_0.1.10     expm_0.999-6
#> [43] codetools_0.2-18       MASS_7.3-57            bitops_1.0-7
#> [46] grid_4.2.1             magrittr_2.0.1         docopt_0.7.1
#> [49] gld_2.6.4              cli_3.3.0              stringi_1.7.6
#> [52] XVector_0.36.0         doParallel_1.0.16      boot_1.3-28
#> [55] HDCI_1.0-2             iterators_1.0.13       tools_4.2.1
#> [58] float_0.3-0            parallel_4.2.1         fastmap_1.1.0
#> [61] survival_3.3-1         yaml_2.2.1             knitr_1.37
```