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 these 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 covariates (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 covariates data contains covariates and confounders with each row per sample and each column per variable. The covariates data has to be numeric or binary. Categorical variables should be converted into binary dummy variables.

- `microbVar`: This takes a single or vector of microbiome variable names (e.g., taxa, OTU and ASV names) of interest. Default is "all" meaning all microbiome variables will be analyzed. If a subset of microbiome variables is specified, the output will only contain the specified variables, and p-value adjustment for multiple testing will only be applied to the subset.
- `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 `0.05`.
- `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.

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. `fdrRate`: FDR rate used for the analysis. `adjust_method`: Multiple testing adjust method used for the analysis.

Example

The example generates an example data from scratch, with 10 taxon, 40 subjects, and 3 covariates.

```
library(IFAA)

set.seed(1)

## create an ID variable for the example data
ID <- seq_len(20)

## generate three covariates x1, x2, and x3, with x2 binary
x1 <- rnorm(20)
x2 <- rbinom(20, 1, 0.5)
x3 <- rnorm(20)
dataC <- data.frame(cbind(ID, x1, x2, x3))

## Coefficients for x1, x2, and x3 among 10 taxa.
beta_1 <- c(0.1, rep(0, 9))
beta_2 <- c(0, 0.2, rep(0, 8))
beta_3 <- rnorm(10)
beta_mat <- cbind(beta_1, beta_2, beta_3)

## Generate absolute abundance for 10 taxa in ecosystem.
dataM_eco <- floor(exp(10 + as.matrix(dataC[, -1]) %*% t(beta_mat) + rnorm(200, sd = 0.05)))

## Generate sequence depth and generate observed abundance
Ci <- runif(20, 0.01, 0.05)
dataM <- floor(apply(dataM_eco, 2, function(x) x * Ci))
colnames(dataM) <- paste0("rawCount", 1:10)

## Randomly introduce 0 to make 25% sparsity level.
```

```r
dataM[sample(seq_len(length(dataM)), length(dataM) / 4)] <- 0

dataM <- data.frame(cbind(ID, dataM))

## The following steps are to create a SummarizedExperiment object.
## If you already have a SummarizedExperiment format data, you can
## ignore the following steps and directly feed it to the IFAA function.

## Merge two dataset by ID variable
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 SummarizedExperiment object
test_dat <- SummarizedExperiment::SummarizedExperiment(
  assays = list(MicrobData = t(dataM_sub)), colData = dataC_sub
)
```

If you already have a SummarizedExperiment format data, you can ignore the above steps for creating a SummarizedExperiment object. In the generated data, rawCount1 is associated with x1, rawCount2 is associated with x2, and the sparsity level is 25%. Next we analyze the data to test the association between microbiome and the variable `"x1"`, `"x2"` while adjusting for the variable (potential confounder) `"x3"`.

```r
set.seed(123) # For full reproducibility
results <- IFAA(
  experiment_dat = test_dat,
  testCov = c("x1", "x2"),
  ctrlCov = c("x3"),
  sampleIDname = "ID",
  fdrRate = 0.05,
  nRef = 2,
  paraJobs = 2
)
#> Data dimensions (after removing missing data if any):
#> 20 samples
#> 10 taxa/OTU/ASV
#> 2 testCov variables in the analysis
#> These are the testCov variables:
#> x1, x2
#> 1 ctrlCov variables in the analysis
#> These are the ctrlCov variables:
#> x3
#> 1 binary covariates in the analysis
#> These are the binary covariates:
#> x2
#> 25 percent of microbiome sequencing reads are zero
#> Start Phase 1 analysis
#> 2 parallel jobs are registered for the analysis.
#> 33 percent of phase 1 analysis has been done and it took 0.24 minutes
#> 2 parallel jobs are registered for the analysis.
#> 100 percent of phase 1 analysis has been done and it took 0.48 minutes
#> Start Phase 2 parameter estimation
```

```
#> 2 parallel jobs are registered for analyzing reference taxa in Phase 2
#> 50 percent of Phase 2 is done and it took 0.181 minutes
#> 2 parallel jobs are registered for analyzing reference taxa in Phase 2
#> Entire Phase 2 parameter estimation done and took 0.351 minutes.
#> The entire analysis took 0.83 minutes
```

Notice that this example used "nRef = 2" and "paraJobs = 2" for the sake of speed in this example. Typically one should use the default values for those two arguments.

In this example, we are only interested in testing the associations with `"x1"` and `"x2"` which is why `testCov=c("x1","x2)`. The variables `"x3"` 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_results
sig_results <- subset(summary_res, sig_ind == TRUE)
sig_results
#> DataFrame with 2 rows and 9 columns
#>         taxon         cov  estimate    SE.est    CI.low     CI.up unadj.p.value
#>   <character> <character> <numeric> <numeric> <numeric> <numeric>     <numeric>
#> 1   rawCount1          x1 0.0778196 0.0204586 0.0377207  0.117918   1.42517e-04
#> 2   rawCount2          x2 0.1842519 0.0387891 0.1082253  0.260278   2.03319e-06
#>   adj.p.value   sig_ind
#>     <numeric> <logical>
#> 1 3.09872e-03      TRUE
#> 2 4.42074e-05      TRUE
```

After adjusting for multiple testing, the results found `"rawCount1"` associated with `"x1"`, and `"rawCount2"` associated with `"x2"`, while adjusting for `"x3"`. The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to β^k in the model equation.

The interpretation is:

- Every unit increase in `"x1"` is associated with approximately 7.8% increase in the absolute abundance of `"rawCount1"` in the entire ecosystem of interest (while controlling for `"x2"` and `"x3"`); Every unit increase in `"x2"` is associated with approximately 18.4% increase in the absolute abundance of `"rawCount2"` in the entire ecosystem of interest (while controlling for `"x1"` and `"x3"`), .

If only interested in certain taxa, say `"rawCount1", "rawCount2", and "rawCount3"`, one can do:

```
set.seed(123) # For full reproducibility

results <- IFAA(
  experiment_dat = test_dat,
  microbVar = c("rawCount1", "rawCount2", "rawCount3"),
  testCov = c("x1", "x2"),
  ctrlCov = c("x3"),
  sampleIDname = "ID",
  fdrRate = 0.05,
  nRef = 2,
  paraJobs = 2
)
```

Again, notice that this example used "nRef = 2" and "paraJobs = 2" for the sake of speed in this example. Typically one should use the default values for those two arguments.

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 four inputs to the function: `experiment_dat`, `microbVar`, `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 covariates (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 covariates data contains covariates and confounders with each row per sample and each column per variable. The covariates data has to be numeric or binary. Categorical variables should be converted into binary dummy variables.
- `microbVar`: This takes a single or vector of microbiome variable names (e.g., taxa, OTU and ASV names) of interest for the numerator of the ratios. Default is "all" meaning all microbiome variables (except denominator) will be analyzed as numerators. If a subset of microbiome variables is specified, the output will only contain the specified ratios, and p-value adjustment for multiple testing will only be applied to the subset.
- `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 `0.05`.
- `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'.

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 a dataset of covariates and confounders used in the analyses, total time used in minutes, FDR rate, and multiple testing adjustment method used.

Examples

The example used data generated from IFAA example, with 10 taxon, 40 subjects, and 3 covariates.

Next we analyze the data to test the associations between the ratios "rawCount5/rawCount10" and "rawCount8/rawCount10" and all the three variables `"x1"`, `"x2"` and `"x3"` in a multivariate model where all `"x1"`, `"x2"` and `"x3"` are independent variables simultaneously.

```
set.seed(123) # For full reproducibility

resultsRatio <- MZILN(
  experiment_dat = test_dat,
  microbVar = c("rawCount5","rawCount8"),
  refTaxa = c("rawCount10"),
  allCov = c("x1", "x2", "x3"),
  sampleIDname = "ID",
  fdrRate = 0.05,
  paraJobs = 2
)
#> Data dimensions (after removing missing data if any):
#> 20 samples
#> 10 taxa/OTU/ASV
#> 3 testCov variables in the analysis
#> These are the testCov variables:
#> x1, x2, x3
#> 0 ctrlCov variables in the analysis
#> 1 binary covariates in the analysis
#> These are the binary covariates:
#> x2
```

```
#> 25 percent of microbiome sequencing reads are zero
#> 2 parallel jobs are registered for analyzing reference taxa in Phase 2
#> Estimation done for the 1th denominator taxon: rawCount10 and it took 0.28 minutes
#> The entire analysis took 0.28 minutes
```

Again, notice that this example used "nRef = 2" and "paraJobs = 2" for the sake of speed in this example. Typically one should use the default values for those two arguments.

The final analysis results can be extracted as follows:

```
summary_res_ratio <- resultsRatio$full_results
summary_res_ratio
#> DataFrame with 6 rows and 10 columns
#>       ref_tax       taxon         cov    estimate    SE.est     CI.low
#>   <character> <character> <character>   <numeric> <numeric>  <numeric>
#> 1  rawCount10   rawCount5          x1  0.00997480 0.0228440 -0.0347994
#> 2  rawCount10   rawCount5          x2 -0.09323061 0.0386687 -0.1690213
#> 3  rawCount10   rawCount5          x3  1.58194615 0.0276923  1.5276693
#> 4  rawCount10   rawCount8          x1 -0.00681531 0.0211971 -0.0483617
#> 5  rawCount10   rawCount8          x2 -0.00445645 0.0394673 -0.0818123
#> 6  rawCount10   rawCount8          x3 -0.91801508 0.0297290 -0.9762839
#>        CI.up unadj.p.value adj.p.value   sig_ind
#>    <numeric>     <numeric>   <numeric> <logical>
#> 1  0.0547490     0.6623657   1.0000000     FALSE
#> 2 -0.0174399     0.0159084   0.0477253      TRUE
#> 3  1.6362230     0.0000000   0.0000000      TRUE
#> 4  0.0347310     0.7478158   1.0000000     FALSE
#> 5  0.0728994     0.9100979   1.0000000     FALSE
#> 6 -0.8597462     0.0000000   0.0000000      TRUE
```

The regression coefficients and their 95% confidence intervals are provided. These coefficients correspond to α^k in the model equation.

The results show that the ratio "rawCount5/rawCount10" is significantly associated with `"x2"` and `"x3"` and the ratio "rawCount8/rawCount10" is significantly associated with `"x3"` after multiple testing adjustment. The interpretation is:

- Every unit increase in `"x2"` is associated with approximately 9.3% reduction in the abundance ratio `"rawCount5/rawCount10"` (while controlling for `"x1"` and `"x3"`); Every unit increase in `"x3"` is associated with approximately 386.4% (=exp(1.582)-1) increase in the abundance ratio `"rawCount5/rawCount10"` (while controlling for `"x1"` and `"x2"`). Every unit increase in `"x3"` is associated with approximately 60.1% (=1-exp(-0.918)) decrease in the abundance ratio `"rawCount8/rawCount10"` (while controlling for `"x1"` and `"x2"`).

If the purpose is to explore all taxa ratios with "rawCount10" being the denominator, one can do:

```
set.seed(123) # For full reproducibility
resultsAllRatio <- MZILN(
  experiment_dat = test_dat,
  refTaxa = c("rawCount10"),
  allCov = c("x1", "x2", "x3"),
  sampleIDname = "ID",
  fdrRate = 0.05,
  paraJobs = 2
)
#> Data dimensions (after removing missing data if any):
#> 20 samples
```

```
#> 10 taxa/OTU/ASV
#> 3 testCov variables in the analysis
#> These are the testCov variables:
#> x1, x2, x3
#> 0 ctrlCov variables in the analysis
#> 1 binary covariates in the analysis
#> These are the binary covariates:
#> x2
#> 25 percent of microbiome sequencing reads are zero
#> 2 parallel jobs are registered for analyzing reference taxa in Phase 2
#> Estimation done for the 1th denominator taxon: rawCount10 and it took 0.31 minutes
#> The entire analysis took 0.31 minutes
```

Again, notice that this example used "nRef = 2" and "paraJobs = 2" for the sake of speed in this example. Typically one should use the default values for those two arguments.

The results for the ratio "rawCount5/rawCount10" can extracted as follows:

```
summary_res_ratio <- resultsAllRatio$full_results
summary_res_ratio[summary_res_ratio$taxon == "rawCount5", , 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  rawCount10   rawCount5          x1  0.0099748 0.0228440 -0.0347994
#> 2  rawCount10   rawCount5          x2 -0.0932306 0.0386687 -0.1690213
#> 3  rawCount10   rawCount5          x3  1.5819462 0.0276923  1.5276693
#>        CI.up unadj.p.value adj.p.value   sig_ind
#>    <numeric>     <numeric>   <numeric> <logical>
#> 1  0.0547490     0.6623657     1.00000     FALSE
#> 2 -0.0174399     0.0159084     0.20252     FALSE
#> 3  1.6362230     0.0000000     0.00000      TRUE
```

The interpretation of the association estimates remain the same. Notice that the significance dropped for the association between the ratio "rawCount5/rawCount10" and `"x2"` due to the correction for multiple testing (i.e., 0.2025>0.05). It is because this exploratory analysis included all microbiome variables and thus the p-value adjustment is more stringent for multiple testing which led to the larger adjusted p-value.

The results for the ratio "rawCount8/rawCount10" can be extracted similarly.

To extract all significant ratios with `"rawCount10"` being the denominator,

one can do: `subset(summary_res_ratio,sig_ind==TRUE)`.

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 4.4.0 beta (2024-04-15 r86425)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 22.04.4 LTS
#>
#> Matrix products: default
#> BLAS:   /home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so
#> 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=en_GB              LC_COLLATE=C
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: America/New_York
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#> [1] IFAA_1.5.2
#>
#> loaded via a namespace (and not attached):
#>  [1] SummarizedExperiment_1.33.3 gld_2.6.6
#>  [3] shape_1.4.6.1               xfun_0.43
#>  [5] Biobase_2.63.1              lattice_0.22-6
#>  [7] mathjaxr_1.6-0              tools_4.4.0
#>  [9] stats4_4.4.0                parallel_4.4.0
#> [11] proxy_0.4-27                Matrix_1.7-0
#> [13] data.table_1.15.4           rngtools_1.5.2
#> [15] S4Vectors_0.41.6            readxl_1.4.3
#> [17] lifecycle_1.0.4             GenomeInfoDbData_1.2.12
#> [19] rootSolve_1.8.2.4           stringr_1.5.1
#> [21] compiler_4.4.0              Exact_3.2
#> [23] RhpcBLASctl_0.23-42         codetools_0.2-20
#> [25] GenomeInfoDb_1.39.14        htmltools_0.5.8.1
#> [27] DescTools_0.99.54           class_7.3-22
#> [29] yaml_2.3.8                  glmnet_4.1-8
#> [31] crayon_1.5.2                MASS_7.3-60.2
#> [33] DelayedArray_0.29.9         doRNG_1.8.6
#> [35] iterators_1.0.14            boot_1.3-30
#> [37] abind_1.4-5                 foreach_1.5.2
#> [39] parallelly_1.37.1           digest_0.6.35
#> [41] stringi_1.8.3               HDCI_1.0-2
#> [43] mvtnorm_1.2-4               slam_0.1-50
#> [45] splines_4.4.0               fastmap_1.1.1
#> [47] grid_4.4.0                  lmom_3.0
#> [49] expm_0.999-9                cli_3.6.2
#> [51] SparseArray_1.3.5           magrittr_2.0.3
#> [53] S4Arrays_1.3.7              survival_3.5-8
#> [55] e1071_1.7-14                withr_3.0.0
#> [57] UCSC.utils_0.99.7           float_0.3-2
#> [59] MatrixExtra_0.1.15          rmarkdown_2.26
#> [61] XVector_0.43.1              httr_1.4.7
#> [63] matrixStats_1.3.0           cellranger_1.1.0
#> [65] evaluate_0.23               knitr_1.46
#> [67] GenomicRanges_1.55.4        IRanges_2.37.1
```

```
#> [69] doParallel_1.0.17          rlang_1.1.3
#> [71] Rcpp_1.0.12                glue_1.7.0
#> [73] BiocGenerics_0.49.1        rstudioapi_0.16.0
#> [75] jsonlite_1.8.8             R6_2.5.1
#> [77] MatrixGenerics_1.15.1      zlibbioc_1.49.3
```