Metabolite identification using MetaboRAID

Motivation

Metabolite identification in untargeted experiments is still a key and often the most difficult data processing step in mass spectrometry-based experiments. Recently there has been a tremendous effort by the community to develop new approaches for the identification of unknowns which has led to the development of various different software packages for annotation unknown features. The two most common approaches for identification are based on in-silico and internal library search. These two approaches are based on MS/MS (MS2) fragmentation pattern. The in-silico approach attempts to reconstruct the molecules based on experimental MS/MS and compare the results to a database. Whereas the library-based approach most deals with comparing MS/MS spectra of the molecule of interest to generated spectra of the standard compound. MetaboRAID deals with the in-silico and attempts to provide a unified platform for performing identification using various search engines.

Pipeline

A typical mass spectrometry data processing often includes a multitude of steps including centroiding, feature detection, annotation, MS/MS processing and identification. In a typical scenario, one would start MS1 data pre-processing and proceed to MS2. In R, two of the famous packages for MS1 quantification and annotation are xcms and CAMERA. After data doing the MS1 pre-processing, it is the time to deal with MS/MS spectra. Different tools require different set of parameters and often different data preparation steps, making time consuming for running the identification analysis. MetaboRAID accepts MS2 spectra and MS1 data pre-treated with xcms and CAMERA and performs MS2 pre-processing and identification.

Here we demonstrate how to perform a simple identification using a small dataset.

Install the package and the tools

In order to use the search engines, they have to be installed. To install the search engines you need to use *install_tools* function. Before running it, make sure that you have the latest conda avaialble. Conda can be installed on Windows, IOS and Linux. When installed, rerun R and start from here.

```r
library(metaboraid)
install_tools()
```

We use conda for installing the packages. An environment will be created and all the required tools will be installed in that environemnt. After that you can continue with your analysis.

MS1 quantification

First load CAMERA and metaboraid and read the MS1 data. **You need to convert you “raw” data to mzML file both for MS1 and MS2**

```r
library(CAMERA)
#> Loading required package: Biobase
#> Loading required package: BiocGenerics
#> Loading required package: parallel
#>
#> Attaching package: 'BiocGenerics'
#> The following objects are masked from 'package:parallel':
```

```
#>
#>     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
#>     clusterExport, clusterMap, parApply, parCapply, parLapply,
#>     parLapplyLB, parRapply, parSapply, parSapplyLB
#> The following objects are masked from 'package:stats':
#>
#>     IQR, mad, sd, var, xtabs
#> The following objects are masked from 'package:base':
#>
#>     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
#>     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
#>     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
#>     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
#>     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
#>     union, unique, unsplit, which.max, which.min
#> Welcome to Bioconductor
#>
#>     Vignettes contain introductory material; view with
#>     'browseVignettes()'. To cite Bioconductor, see
#>     'citation("Biobase")', and for packages 'citation("pkgname")'.
#> Loading required package: xcms
#> Loading required package: BiocParallel
#> Loading required package: MSnbase
#> Loading required package: mzR
#> Loading required package: Rcpp
#> Warning in fun(libname, pkgname): mzR has been built against a different Rcpp version
      (1.0.5)
#> than is installed on your system (1.0.6). This might lead to errors
#> when loading mzR. If you encounter such issues, please send a report,
#> including the output of sessionInfo() to the Bioc support forum at
#> https://support.bioconductor.org/. For details see also
#> https://github.com/sneumann/mzR/wiki/mzR-Rcpp-compiler-linker-issue.
#> Loading required package: S4Vectors
#> Loading required package: stats4
#>
#> Attaching package: 'S4Vectors'
#> The following object is masked from 'package:base':
#>
#>     expand.grid
#> Loading required package: ProtGenerics
#>
#> Attaching package: 'ProtGenerics'
#> The following object is masked from 'package:stats':
#>
#>     smooth
#>
#> This is MSnbase version 2.16.1
#>   Visit https://lgatto.github.io/MSnbase/ to get started.
#>
#> Attaching package: 'MSnbase'
#> The following object is masked from 'package:base':
#>
#>     trimws
#>
#> This is xcms version 3.12.0
#>
#> Attaching package: 'xcms'
#> The following object is masked from 'package:stats':
#>
#>     sigma
library(metaboraid)
#> Warning: replacing previous import 'MSnbase::reduce' by 'intervals::reduce' when
#> loading 'metaboraid'
#> Warning: replacing previous import 'intervals::empty' by 'plyr::empty' when
#> loading 'metaboraid'
#> Warning: replacing previous import 'intervals::tail' by 'utils::tail' when
#> loading 'metaboraid'
#> Warning: replacing previous import 'intervals::head' by 'utils::head' when
#> loading 'metaboraid'
#> Warning: replacing previous import 'utils::unzip' by 'zip::unzip' when loading
#> 'metaboraid'
#> Warning: replacing previous import 'utils::zip' by 'zip::zip' when loading
#> 'metaboraid'


# Read MS1
ms1_files<-system.file("ms1data",c("X1_Rep1.mzML","X2_Rep1.mzML"),package = "metaboraid")

print(ms1_files)
#> [1] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms1data/X1_Rep1.mzML

#> [2] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms1data/X2_Rep1.mzML
```

We can now go ahead and do a typical xcms and CAMERA data pre-processing which includes:

1. Mass trace detection
2. Mass trace grouping
3. Converting to CAMERA object
4. Grouping mass tracing into PC groups
5. Finding Isoptopes
6. Rearrange the grouping
7. Adduct detection

```
# 1. mass trace detection
xs <- xcmsSet(ms1_files,method="centWave",ppm=30,peakwidth=c(5,10))

# 2. mass trace matching
xsg <- group(xs)
#> Processing 6638 mz slices ... OK

# 3. convert to CAMERA
xsa <- xsAnnotate(xsg)

# 4. Group mass traces
anF <- groupFWHM(xsa, perfwhm = 0.6)

# 5. Detect isotopes
anI <- findIsotopes(anF, mzabs = 0.01)

# 6. Group using correlation
anIC <- groupCorr(anI, cor_eic_th = 0.75)

# 7. Find adducts
anFA <- findAdducts(anIC, polarity="positive")
```

At this stage we have the quantification analysis done. We are ready to proceed to the identification step. **Please note that, steps 5 and 7 are required in the next steps. So make sure that you have done the isotope and adduct detection.**

MS2 loading and mapping

Now it is the time to load our MS2 data. Our *ms2_files* variable contains a vector of paths for the location of MS2 mzML files.

```
# Read MS2 data
ms2_files<-system.file("ms2data",c("sample1.mzML","sample2.mzML"),package = "metaboraid")

print(ms2_files)
#> [1] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms2data/sample1.mzML

#> [2] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms2data/sample2.mzML
```

Where we have our file paths in *ms2_files* we can use *mapped_features* function to read MS2 files and map the MS2 parent ions onto the features detected by xcms/CAMERA. We do that to be able to later detect adducts and also merge related MS2 peaks.

Here we set mz and retention time deviation for mapping to 10 ppm and 10 seconds.

```
mapped_features<-map_features(inputMS2s = ms2_files,input_camera = anFA,ppm = 10,rt = 10)
```

After this step we have all the mapped and unmapped MS2s in *mapped_features*. This variable is a simple nested list. With two names:

```
names(mapped_features)
#> [1] "mapped"   "unmapped"
```

The *mapped* element is a nested list in which all the MS2s that had a corresponding feature in MS1 data are stored.

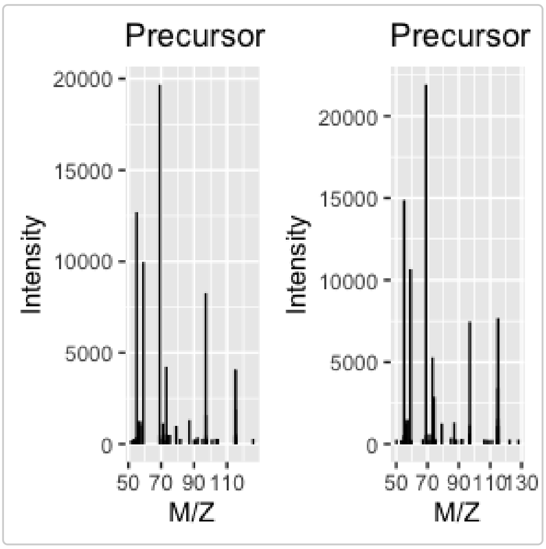
The name of each element of the first level correspond to the index of a feature in the CAMERA object:

```
CAMERA_peaks<-getPeaklist(anFA)

all_index<-names(mapped_features$mapped)

print(CAMERA_peaks[as.numeric(all_index[6]),])
#>           mz    mzmin    mzmax       rt    rtmin  rtmax npeaks ms1data X1_Rep1
#> 132 115.0505 115.0374 115.1233 43.86738 9.863749 137.41     30       2 7907389
#>     X2_Rep1 isotopes adduct pcgroup
#> 132 2842564                     824
```

The second level list contains all the MS2s for a particular feature:

```
print(mapped_features$mapped[[6]])
#> [[1]]
#> Object of class "Spectrum2"
#>  Precursor: 115.0505
#>  Retention time: 0:10
#>  Charge: 1
#>  MSn level: 2
#>  Peaks count: 42
#>  Total ion count: 82889.79
#>
#> [[2]]
#> Object of class "Spectrum2"
#>  Precursor: 115.0505
#>  Retention time: 0:11
#>  Charge: 1
#>  MSn level: 2
#>  Peaks count: 50
#>  Total ion count: 97229.15

plots_MS2s<-lapply(mapped_features$mapped[[6]],plot,plot=F)
```

```
cowplot::plot_grid(plotlist = plots_MS2s)
```



At this stage we have the quantification analysis done. We are ready to proceed to the identification step. **Please note that, steps 5 and 7 are required in the next steps. So make sure that you have done the isotope and adduct detection.**

MS2 loading and mapping

Now it is the time to load our MS2 data. Our *ms2_files* variable contains a vector of paths for the location of MS2 mzML files.

```
# Read MS2 data
ms2_files<-system.file("ms2data",c("sample1.mzML","sample2.mzML"),package = "metaboraid")

print(ms2_files)
#> [1] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms2data/sample1.mzML

#> [2] "/Library/Frameworks/R.framework/Versions/4.0/Resources/library/metaboraid/ms2data/sample2.mzML
```

Where we have our file paths in *ms2_files* we can use *mapped_features* function to read MS2 files and map the MS2 parent ions onto the features detected by xcms/CAMERA. We do that to be able to later detect adducts and also merge related MS2 peaks.

Here we set mz and retention time deviation for mapping to 10 ppm and 10 seconds.

```
mapped_features<-map_features(inputMS2s = ms2_files,input_camera = anFA,ppm = 10,rt = 10)
```

After this step we have all the mapped and unmapped MS2s in *mapped_features*. This variable is a simple nested list. With two names:

```
names(mapped_features)
#> [1] "mapped"   "unmapped"
```

The *mapped* element is a nested list in which all the MS2s that had a corresponding feature in MS1 data are stored.

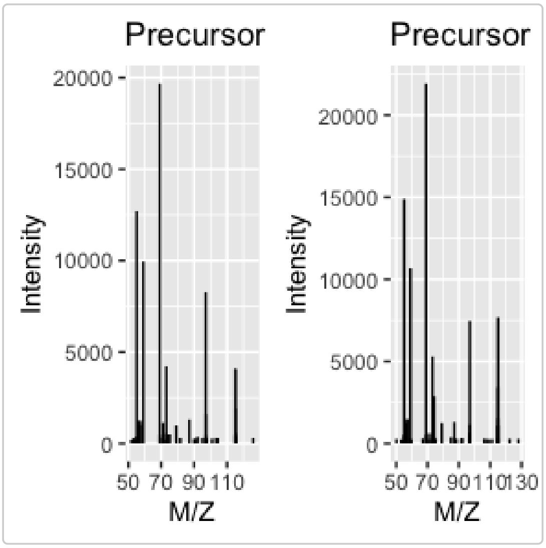
The name of each element of the first level correspond to the index of a feature in the CAMERA object:

```
CAMERA_peaks<-getPeaklist(anFA)
```

```
all_index<-names(mapped_features$mapped)

print(CAMERA_peaks[as.numeric(all_index[6]),])
#>           mz    mzmin    mzmax       rt    rtmin  rtmax npeaks ms1data X1_Rep1
#> 132 115.0505 115.0374 115.1233 43.86738 9.863749 137.41     30       2 7907389
#>     X2_Rep1 isotopes adduct pcgroup
#> 132 2842564                     824
```

The second level list contains all the MS2s for a particular feature:

```
print(mapped_features$mapped[[6]])
#> [[1]]
#> Object of class "Spectrum2"
#>  Precursor: 115.0505
#>  Retention time: 0:10
#>  Charge: 1
#>  MSn level: 2
#>  Peaks count: 42
#>  Total ion count: 82889.79
#>
#> [[2]]
#> Object of class "Spectrum2"
#>  Precursor: 115.0505
#>  Retention time: 0:11
#>  Charge: 1
#>  MSn level: 2
#>  Peaks count: 50
#>  Total ion count: 97229.15

plots_MS2s<-lapply(mapped_features$mapped[[6]],plot,plot=F)

cowplot::plot_grid(plotlist = plots_MS2s)
```



So this particular MS1 feature has two MS2s associated with it. The unmapped element of the list contains all other MS2s that did not get a feature.

Pre-process MS2 data (optional)

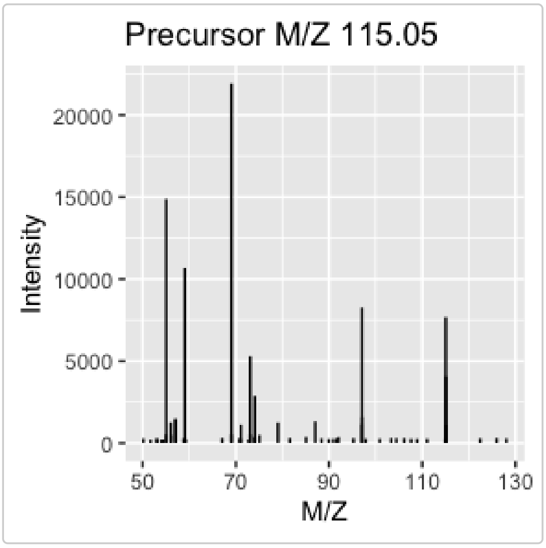
MS2 data can be noisy and sparse. Often multiple ions from the same precursor are picked by the instrument to fragment. MetaboRAID has the option to merge and pre-process the mapped ions.

If order to do merge, we use *preprocess_msms* which provides the centroiding and merging options both for mapped and unmapped ions.

```
mapped_features_merge<-preprocess_msms(mapped_features,centroid = F,merge =
        T,centroid_after_merge = F,ppm = 10,ppm_precursor = 5,rt = 10,abs_mz_precursor =
        0.01,abs_mz = 0.05)
```

We can see that the spectra is now merge and contains more peaks but the redundant ones have been merge into single peaks.

```
plots_MS2s_merge<-lapply(mapped_features_merge$mapped[[6]],plot,plot=F)

cowplot::plot_grid(plotlist = plots_MS2s_merge)
```



At this stage we could use *plots_MS2s_merge* instead of *mapped_features* throughout the rest of analysis but we choose to do go with *mapped_features* just to show the standard behaviour of the package.

Neutral mass estimation

After this step we go ahead and perform neutral mass estimation. We do that to decrease the search space of the proceeding search engines. The way that neutral mass estimation works is to first find out if a MS2 has a feature and that feature is an adduct or has been annotated with isotope information by CAMERA. If so, the neutral mass is deduced by taking the adduct and isotope information into account. If the MS2 does not have adducts or not even a feature, depending on the user choice we generate a set of possible adducts for this MS2 So effectively the MS2 will be search multiple times.

All of these are achieved using *mapped_adducts* function. In this function, we need to input the list of MS2s as generated by *mapped_features*, CAMERA quantification object, precursor and fragment mass devation as well as ionization, addcut rules and other parameters that later will be used by the search engine to perform the search. So this is important to note that the deviation parameters will be used by search engines not for mapping.

Two important parameters here are *searchMultipleChargeAdducts* which force MetaboRAID to generate multiple adducts for a MS2 if it we did not find an adduct for it. The other parameter to consider is outputDir, you need to make sure that you set this parameter everytime to a different directory so the result of the previous run won’t be overwritten.

```
mapped_adducts<-map_adducts(inputMS2List=mapped_features,input_camera=anFA,
                     precursorppm=10,
                    fragmentppm=20,fragmentabs=0.01,minPrecursorMass=NA,maxPrecursorMass=NA,
                     minPeaks=10,maxSpectra=10,mode="pos",adductRules="primary",
                    outputDir="general_parameters",searchMultipleChargeAdducts=T,
                     includeMapped=T,includeUnmapped=F,verbose=F)
```

This function will generate a file called *parameter_files.zip* which can be passed to all the search engines. This files contains all the parameters that generated and put in *outputDir*.

Identification

We are now ready to perform the identification. We demostrate CSI-FINGERID here:

```
#> Loading required package: foreach
#> Loading required package: future
#> 
#> Attaching package: 'future'
#> The following object is masked from 'package:S4Vectors':
#> 
#>     values
```

The results is a data.frame of CSI-FINGERID results:

We can instead also run another search engine. Let's try the above using Metfrag:

Mapping to quantification

The last stage of the processing is to map identified metabolites to quantification data. We can do this using *map_to_camera* function. This function accepts the scores and CAMERA object as well as mass and retention time deviation and maps the identification results to quantification.

```
final_results<-metaboraid::map_to_camera(input_camera = anFA,identification_result =
        ID_results_metfrag,ppm = 10,rt = 10,higher_the_better = T,score_column =
        "OfflineMetFusionScore",impute = F,mapping_method = "fast")
```

The results of this function is a list with two elements:

```
names(final_results)
#> [1] "peakMatrix"   "variableData"
```

The *peakMatrix* contains the quantification from CAMERA

```
head(final_results$peakMatrix)
#>   dataMatrix       mz    mzmin    mzmax        rt     rtmin     rtmax npeaks
#> 1 variable_1 70.01312 70.01310 70.01315  30.14575  30.07937  30.21213      2
#> 2 variable_2 70.98003 70.98003 70.98003  29.81187  29.81187  29.81187      1
#> 3 variable_3 71.04525 71.02962 71.08566  25.75537  20.89661  30.21213      4
#> 4 variable_4 71.53019 71.52999 71.53039  29.81187  29.81187  29.81187      2
#> 5 variable_5 72.00959 72.00959 72.00959  29.81187  29.81187  29.81187      1
#> 6 variable_6 72.08143 72.08142 72.08144 298.20779 298.12835 298.28722      2
#>   ms1data    X1_Rep1   X2_Rep1 isotopes            adduct pcgroup
#> 1       2 5379468.14 3865614.4                                  9
#> 2       1   75548.21        NA                                  9
#> 3       2   96515.02  918107.0                               2218
#> 4       1  203144.29        NA          [M+H-CO]+ 98.5223       9
#> 5       1  108159.77        NA            [M+Na]+ 49.0176       9
#> 6       2  795447.68  477918.4                               1089
```

And *variableData* data contains the identification results.

```
head(final_results$variableData)
#>      variableMetadata Score     MonoisotopicMass SMILES    InChIKey
#> [1,] "variable_1"     "Unknown" "Unknown"        "Unknown" "Unknown"
#> [2,] "variable_2"     "Unknown" "Unknown"        "Unknown" "Unknown"
#> [3,] "variable_3"     "Unknown" "Unknown"        "Unknown" "Unknown"
#> [4,] "variable_4"     "Unknown" "Unknown"        "Unknown" "Unknown"
#> [5,] "variable_5"     "Unknown" "Unknown"        "Unknown" "Unknown"
#> [6,] "variable_6"     "Unknown" "Unknown"        "Unknown" "Unknown"
#>      NoExplPeaks NumberPeaksUsed InChI     OfflineMetFusionScore
#> [1,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#> [2,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#> [3,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#> [4,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#> [5,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#> [6,] "Unknown"   "Unknown"       "Unknown" "Unknown"
#>      MaximumTreeDepth Identifier ExplPeaks InChIKey2 InChIKey1 IUPACName
#> [1,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#> [2,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#> [3,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#> [4,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#> [5,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#> [6,] "Unknown"        "Unknown"  "Unknown" "Unknown" "Unknown" "Unknown"
#>      FragmenterScore MolecularFormula FragmenterScore_Values
#> [1,] "Unknown"       "Unknown"        "Unknown"
#> [2,] "Unknown"       "Unknown"        "Unknown"
#> [3,] "Unknown"       "Unknown"        "Unknown"
#> [4,] "Unknown"       "Unknown"        "Unknown"
#> [5,] "Unknown"       "Unknown"        "Unknown"
#> [6,] "Unknown"       "Unknown"        "Unknown"
#>      FormulasOfExplPeaks XlogP3    parentRT  parentMZ  fileName
#> [1,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#> [2,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#> [3,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#> [4,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#> [5,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#> [6,] "Unknown"           "Unknown" "Unknown" "Unknown" "Unknown"
#>      xcmsCamera_mz      xcmsCamera_mzmin   xcmsCamera_mzmax   xcmsCamera_rt
#> [1,] "70.0131241846331" "70.0131017207328" "70.0131466485334" "30.1457526"
#> [2,] "70.980026851109"  "70.980026851109"  "70.980026851109"  "29.81187"
#> [3,] "71.045252437376"  "71.0296180575328" "71.0856591098561" "25.755372"
#> [4,] "71.5301938409496" "71.5299942307591" "71.5303934511402" "29.81187"
#> [5,] "72.0095941708764" "72.0095941708764" "72.0095941708764" "29.81187"
#> [6,] "72.0814275617967" "72.0814150386057" "72.0814400849877" "298.207785"
#>      xcmsCamera_rtmin xcmsCamera_rtmax xcmsCamera_npeaks xcmsCamera_isotopes
#> [1,] "30.079374"      "30.2121312"     "2"               ""
#> [2,] "29.81187"       "29.81187"       "1"               ""
#> [3,] "20.8966098"     "30.2121312"     "4"               ""
#> [4,] "29.81187"       "29.81187"       "2"               ""
#> [5,] "29.81187"       "29.81187"       "1"               ""
#> [6,] "298.128348"     "298.287222"     "2"               ""
#>      xcmsCamera_adduct   xcmsCamera_pcgroup imputed
#> [1,] ""                  "9"                "No"
#> [2,] ""                  "9"                "No"
#> [3,] ""                  "2218"             "No"
#> [4,] "[M+H-CO]+ 98.5223" "9"                "No"
#> [5,] "[M+Na]+ 49.0176"   "9"                "No"
#> [6,] ""                  "1089"             "No"
```

The first columns of the both matrices are the key to match these two dataframes.

```
all(final_results$peakMatrix[,1]==final_results$variableData[,1])
#> [1] TRUE
```

At this stage one can go ahead with the statistical or pathway analysis depending on specific needs of the project.

```
sessionInfo()
#> R version 4.0.4 (2021-02-15)
#> Platform: x86_64-apple-darwin17.0 (64-bit)
#> Running under: macOS Big Sur 10.16
#>
#> Matrix products: default
#> BLAS:   /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRblas.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/4.0/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#> [1] stats4    parallel  stats     graphics  grDevices utils     datasets
#> [8] methods   base
#>
#> other attached packages:
#>  [1] future.apply_1.7.0  doFuture_0.12.0     future_1.21.0
#>  [4] foreach_1.5.1       metaboraid_1.0.0.0  CAMERA_1.46.0
#>  [7] xcms_3.12.0         MSnbase_2.16.1      ProtGenerics_1.22.0
#> [10] S4Vectors_0.28.1    mzR_2.24.1          Rcpp_1.0.6
#> [13] BiocParallel_1.24.1 Biobase_2.50.0      BiocGenerics_0.36.0
#>
#> loaded via a namespace (and not attached):
#>   [1] colorspace_2.0-0           ellipsis_0.3.1
#>   [3] htmlTable_2.1.0            XVector_0.30.0
#>   [5] GenomicRanges_1.42.0       base64enc_0.1-3
#>   [7] rstudioapi_0.13            farver_2.1.0
#>   [9] listenv_0.8.0              affyio_1.60.0
#>  [11] fansi_0.4.2                codetools_0.2-18
#>  [13] splines_4.0.4              ncdf4_1.17
#>  [15] doParallel_1.0.16          impute_1.64.0
#>  [17] robustbase_0.93-7          knitr_1.31
#>  [19] Formula_1.2-4              jsonlite_1.7.2
#>  [21] cluster_2.1.0              vsn_3.58.0
#>  [23] png_0.1-7                  graph_1.68.0
#>  [25] BiocManager_1.30.10        compiler_4.0.4
#>  [27] backports_1.2.1            assertthat_0.2.1
#>  [29] Matrix_1.3-2               limma_3.46.0
#>  [31] htmltools_0.5.1.1          tools_4.0.4
#>  [33] igraph_1.2.6               gtable_0.3.0
#>  [35] glue_1.4.2                 GenomeInfoDbData_1.2.4
#>  [37] affy_1.68.0                RANN_2.6.1
#>  [39] dplyr_1.0.5                MALDIquant_1.19.3
#>  [41] jquerylib_0.1.3            vctrs_0.3.6
#>  [43] preprocessCore_1.52.1      progressr_0.7.0
#>  [45] iterators_1.0.13           xfun_0.21
#>  [47] stringr_1.4.0              globals_0.14.0
#>  [49] lifecycle_1.0.0            XML_3.99-0.5
#>  [51] DEoptimR_1.0-8             zlibbioc_1.36.0
#>  [53] MASS_7.3-53                scales_1.1.1
#>  [55] pcaMethods_1.82.0          MatrixGenerics_1.2.1
#>  [57] SummarizedExperiment_1.20.0 RBGL_1.66.0
#>  [59] MassSpecWavelet_1.56.0     RColorBrewer_1.1-2
#>  [61] yaml_2.2.1                 reticulate_1.18
#>  [63] gridExtra_2.3              ggplot2_3.3.3
#>  [65] sass_0.3.1                 rpart_4.1-15
#>  [67] latticeExtra_0.6-29        stringi_1.5.3
#>  [69] highr_0.8                  checkmate_2.0.0
#>  [71] zip_2.1.1                  GenomeInfoDb_1.26.2
#>  [73] intervals_0.15.2           rlang_0.4.10
#>  [75] pkgconfig_2.0.3            matrixStats_0.58.0
#>  [77] bitops_1.0-6               mzID_1.28.0
#>  [79] evaluate_0.14              lattice_0.20-41
#>  [81] purrr_0.3.4                labeling_0.4.2
#>  [83] htmlwidgets_1.5.3          cowplot_1.1.1
#>  [85] tidyselect_1.1.0           parallelly_1.24.0
#>  [87] plyr_1.8.6                 magrittr_2.0.1
#>  [89] R6_2.5.0                   IRanges_2.24.1
#>  [91] generics_0.1.0             Hmisc_4.5-0
#>  [93] DelayedArray_0.16.2        DBI_1.1.1
#>  [95] pillar_1.5.1               foreign_0.8-81
#>  [97] MsCoreUtils_1.2.0          survival_3.2-7
#>  [99] RCurl_1.98-1.2             nnet_7.3-15
#> [101] tibble_3.1.0               crayon_1.4.1
#> [103] utf8_1.1.4                 rmarkdown_2.7
#> [105] jpeg_0.1-8.1               grid_4.0.4
#> [107] data.table_1.14.0          digest_0.6.27
#> [109] munsell_0.5.0              bslib_0.2.4
```