Workflow metabolite set enrichment analysis with bootstrapping with `bmetenrichr`

This R-package aims to perform metabolite set enrichment analysis (MSEA) on single-cell metabolomics datasets. In contrast to bulk-metabolomics, metabolite annotation is often more ambiguous with fully resolved molecular structures. That means, annotations are vectors of isomeric (and/or isobaric) molecules, complicating downstream MSEA. This package uses a boostrapping approach by performing enrichment analyses many times with random sampling of the isomers/isobars.

```
options(stringsAsFactors = FALSE, warn = -1)

## install devtools if not installed
if(!("devtools" %in% rownames(installed.packages()))){
  install.packages("devtools",  repos = c(CRAN = "http://cran.rstudio.com"))
  }

## install bmetenrichr if not installed
if(!("bmetenrichr" %in% rownames(installed.packages()))){
   devtools::install_github(repo = "martijnmolenaar/bmetenrichr", build_vignettes = TRUE)
}

library(bmetenrichr)
#> Loading required package: ggplot2
#> Loading required package: dplyr
#>
#> Attaching package: 'dplyr'
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
```

The package contains example data from Rappez et al., 2021 (https://doi.org/10.1038/s41592-021-01198-0).

```
data("Rappez_et_al")

## the main input is a single-cell metabolomics matrix with molecules as rows and cells as columns
Rappez_et_al$sc_matrix[1:10,1:10]
#>                          8        15         18         25         27
#> C10H10N4O.K     0.01145815 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H10O6.K      0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H11N4O7P.Na  0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H11N5O3.H    0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H12ClN5O3.K  0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H12ClN5O3.Na 0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#> C10H12ClNO4.H   0.01707534 0.0620677 0.05203464 0.01376432 0.02417577
#> C10H12ClNO4.K   0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
```

```
#> C10H12ClNO4.Na  0.17496077 0.1702377 0.17311070 0.16992632 0.01635700
#> C10H12FN5O4.H   0.00000000 0.0000000 0.00000000 0.00000000 0.00000000
#>                         29          35         38          41          47
#> C10H10N4O.K     0.008558847 0.000000000 0.00000000 0.000000000 0.000000000
#> C10H10O6.K      0.000000000 0.000000000 0.00000000 0.000000000 0.006726224
#> C10H11N4O7P.Na  0.000000000 0.005782876 0.02968958 0.000000000 0.000000000
#> C10H11N5O3.H    0.000000000 0.000000000 0.00000000 0.000000000 0.000000000
#> C10H12ClN5O3.K  0.000000000 0.000000000 0.00000000 0.000000000 0.000000000
#> C10H12ClN5O3.Na 0.000000000 0.000000000 0.00000000 0.000000000 0.000000000
#> C10H12ClNO4.H   0.028755533 0.030977567 0.03611660 0.009692678 0.031959898
#> C10H12ClNO4.K   0.000000000 0.000000000 0.00000000 0.000000000 0.000000000
#> C10H12ClNO4.Na  0.174631645 0.147417684 0.12427280 0.160955912 0.120047020
#> C10H12FN5O4.H   0.000000000 0.000000000 0.00000000 0.000000000 0.000000000

## for the molecules, a vector with molecular formulas plus adduct is required
rownames(Rappez_et_al$sc_matrix)[1:10]
#>  [1] "C10H10N4O.K"     "C10H10O6.K"      "C10H11N4O7P.Na"  "C10H11N5O3.H"
#>  [5] "C10H12ClN5O3.K"  "C10H12ClN5O3.Na" "C10H12ClNO4.H"   "C10H12ClNO4.K"
#>  [9] "C10H12ClNO4.Na"  "C10H12FN5O4.H"

## a conditions vector is required to define to which condition a given cell belongs
Rappez_et_al$conditions[1:10]
#>  [1] "F" "F" "F" "F" "F" "F" "F" "F" "F" "F"

## in this analysis, only specific annotations should be included as others are extracellular
Rappez_et_al$cellular[1:10]
#>     C10H10N4O.K      C10H10O6.K  C10H11N4O7P.Na    C10H11N5O3.H  C10H12ClN5O3.K
#>           FALSE           FALSE           FALSE            TRUE           FALSE
#> C10H12ClN5O3.Na   C10H12ClNO4.H   C10H12ClNO4.K  C10H12ClNO4.Na   C10H12FN5O4.H
#>           FALSE           FALSE           FALSE           FALSE           FALSE
```

enrichment analysis with isomers

The main enrichment object can be generated as follows. By default, `bmetenrichr` uses LION (Molenaar et al., 2019, GigaScience, https://doi.org/10.1093/gigascience/giz061) as metabolite sets.

```
myTestRun <-
  initEnrichment(scmatrix = Rappez_et_al$sc_matrix,
                 annotations = rownames(Rappez_et_al$sc_matrix),
                 conditions = Rappez_et_al$conditions,
                 include = Rappez_et_al$cellular,
                 condition.x = "U",
                 condition.y = "F"                    )
#>
#> Parsing isomers...
#> single-cell metabolomics matrix of 3385 metabolites and 8807 cells
#> active pathway: LION
#>
#> conditions: F, FI, FIT, U
#>
#> condition.x: U
#> condition.y: F
```

First, metabolites are ranked by `rankScore()`

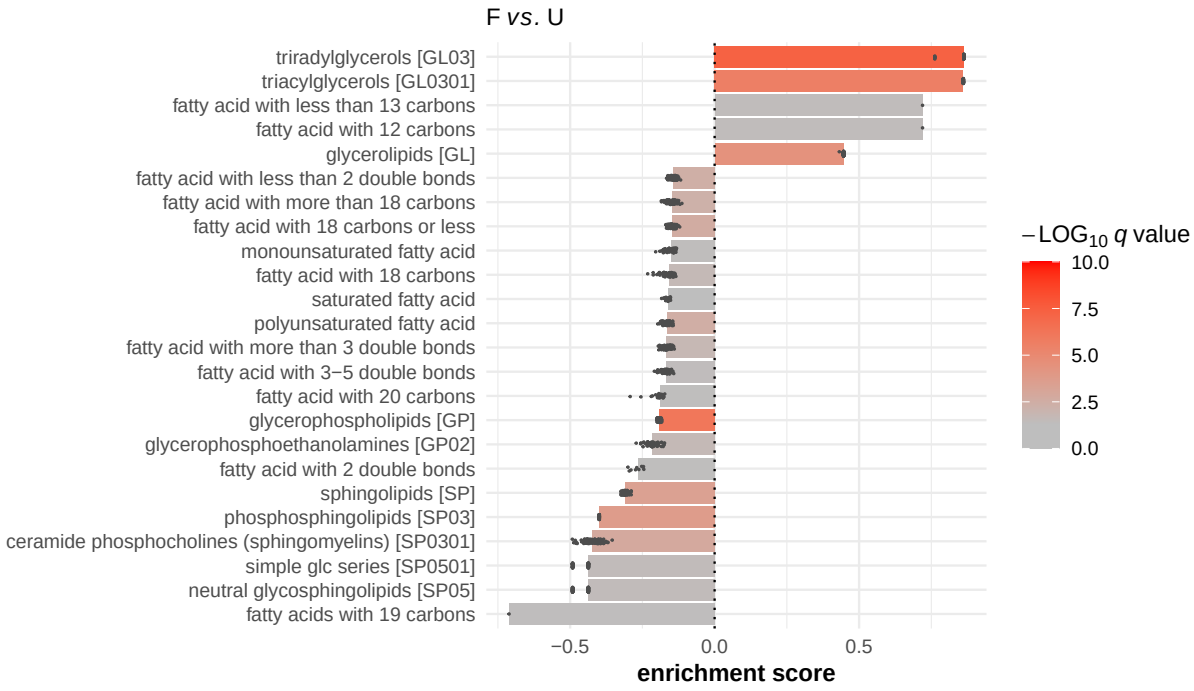
```
## rank metabolites, in this case by t.test statistic

myTestRun <- rankScore(myTestRun, ranking.by = 't.test')
#> number of ties: 19 (1.73%)
```

Then, perform enrichment analysis with n = 100 bootstraps

```
myTestRun <- calcEnrichment(myTestRun, n = 100)
#>
#> Bootstrapping...
#>
#> Match to pathway...
#> 35.43% of annotations were matched to pathway
#>
#> Perform enrichment analysis...
```

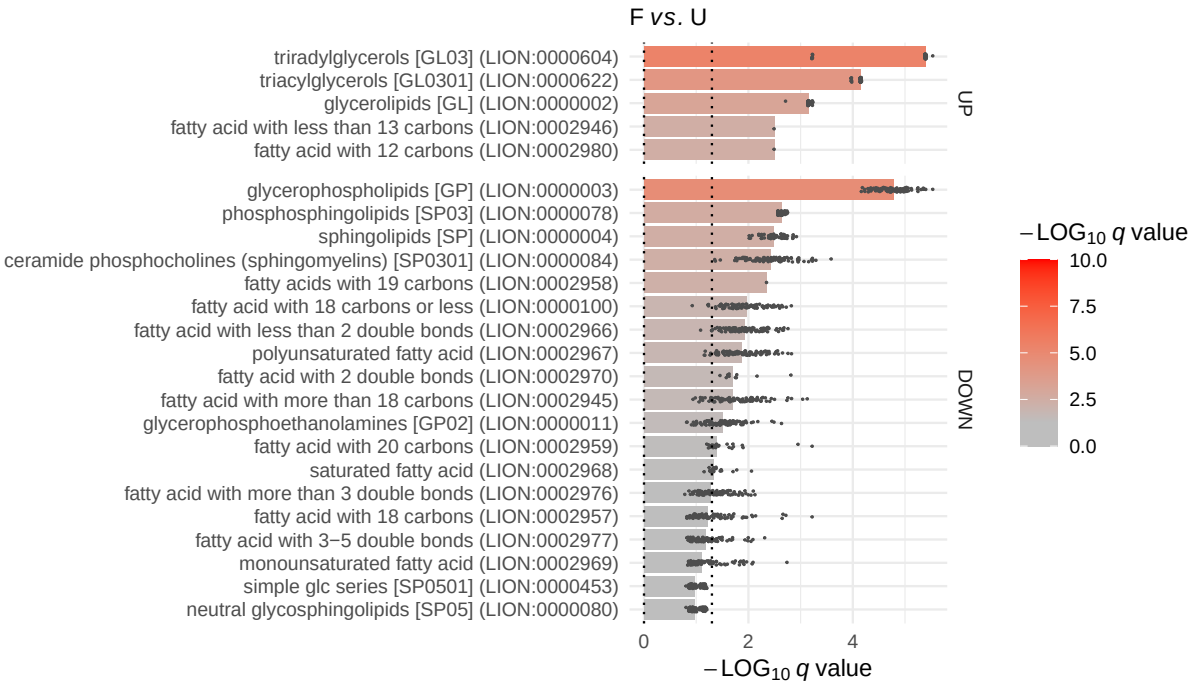
The enrichment analysis can be visualized with `plotEnrichment()`, here with enrichment score (ES) on x-axis

```
plotEnrichment(myTestRun, min.annotations = 5, q.value.cutoff = .05, by.statistic = "ES")
```



Plots can also be arranged with q.values on x-axis, and with LION IDs

```
plotEnrichment(myTestRun, min.annotations = 5, q.value.cutoff = .05, plotIDs = T,
               by.statistic = "q.value")
```



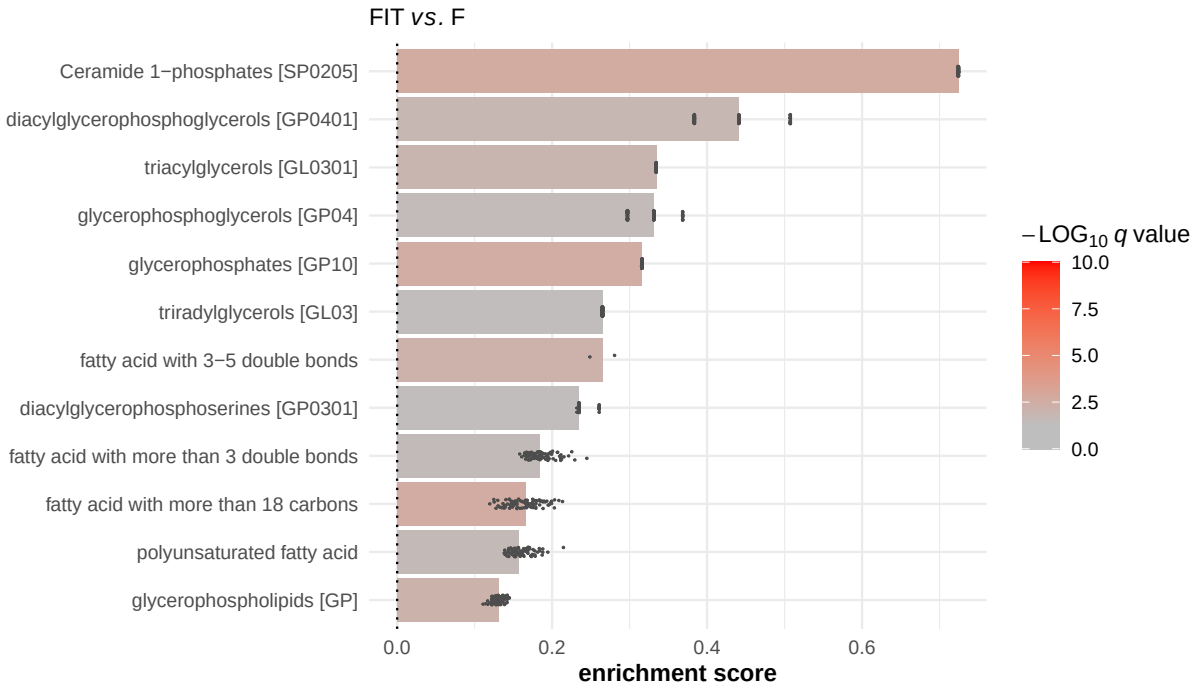
The enrichment analysis can also be exported as `data.frame`:

```
enrichmentTable(myTestRun)[1:10,]
#>         LION_ID                                    LION_name     n  ES_median
#> 1  LION:0000002                          glycerolipids [GL]  28.0  0.4463952
#> 2  LION:0000003                 glycerophospholipids [GP] 290.0 -0.1906772
#> 3  LION:0000004                          sphingolipids [SP]  48.0 -0.3102381
#> 4  LION:0000010             glycerophosphocholines [GP01]  54.0 -0.1780171
#> 5  LION:0000011         glycerophosphoethanolamines [GP02]  69.0 -0.2159125
#> 6  LION:0000012             glycerophosphoinositols [GP06]  49.0 -0.1936927
#> 7  LION:0000013               glycerophosphoserines [GP03]  50.0 -0.1956107
#> 8  LION:0000014             glycerophosphoglycerols [GP04]  25.5 -0.2098761
#> 9  LION:0000030      diacylglycerophosphocholines [GP0101]  39.0 -0.2154537
#> 10 LION:0000038 diacylglycerophosphoethanolamines [GP0201]  47.0 -0.1994681
#>          ES_sd p.value_median   p.value_sd q.value_median   q.value_sd
#> 1  0.001471141   3.787071e-05 7.132690e-06   7.086050e-04 1.293412e-04
#> 2  0.003951007   3.672013e-07 3.060791e-07   1.696656e-05 1.485968e-05
#> 3  0.007722439   2.908494e-04 1.812253e-04   3.241739e-03 1.719838e-03
#> 4  0.023993550   6.907200e-02 6.265088e-02   1.915757e-01 1.107633e-01
#> 5  0.019120498   4.815653e-03 1.342768e-02   3.217896e-02 4.046680e-02
#> 6  0.000000000   5.963835e-02 0.000000e+00   1.769271e-01 1.418379e-02
#> 7  0.009880893   5.187168e-02 2.066770e-02   1.769271e-01 4.052363e-02
#> 8  0.019146433   2.637109e-01 1.088695e-01   4.562688e-01 1.130974e-01
#> 9  0.031850236   7.204803e-02 6.530892e-02   2.013090e-01 1.187774e-01
#> 10 0.017338806   4.831760e-02 4.460437e-02   1.687179e-01 8.174372e-02
```

To compare other conditions, use `setConditions()`:

```
## now, let's test FIT vs F

myTestRun <-  setConditions(object = myTestRun, condition.x = 'F', condition.y = 'FIT')
```

```
myTestRun
#> single-cell metabolomics matrix of 3385 metabolites and 8807 cells
#> active pathway: LION
#>
#> conditions: F, FI, FIT, U
#>
#> condition.x: F
#> condition.y: FIT

## rank metabolites, in this case by t.test statistic

myTestRun <- rankScore(myTestRun, ranking.by = 't.test')
#> number of ties: 24 (2.19%)

## and perform enrichment analysis

myTestRun <- calcEnrichment(myTestRun, n = 100)
#>
#> Bootstrapping...
#>
#> Match to pathway...
#> 34.52% of annotations were matched to pathway
#>
#> Perform enrichment analysis...

plotEnrichment(myTestRun, min.annotations = 5, q.value.cutoff = .05)
```



enrichment analysis with isomers and isobars

By default, only isomers are included. With isobars = TRUE, it's also possible to include isobars within a set m/z range:

```r
## create object

myTestRun <-
  initEnrichment(scmatrix = Rappez_et_al$sc_matrix,
                 isobars = TRUE,                      ## to include isobars (default is FALSE)
                 mass_range_ppm = 3,                  ## mass range to define isobars
                 polarization_mode = "positive",      ## mode is important to include the right adducts
                 annotations = rownames(Rappez_et_al$sc_matrix),
                 conditions = Rappez_et_al$conditions,
                 include = Rappez_et_al$cellular,
                 condition.x = "U",
                 condition.y = "F"                    )
#> polarization_mode is: positive
#>
#> Parsing isomers...
#> Parsing potential isobars...
#>
#> single-cell metabolomics matrix of 3385 metabolites and 8807 cells
#> active pathway: LION
#>
#> conditions: F, FI, FIT, U
#>
#> condition.x: U
#> condition.y: F
```

Downstream, the same workflow can be used:

```r
## rank metabolites, in this case by t.test statistic

myTestRun <- rankScore(myTestRun, ranking.by = 't.test')
#> number of ties: 19 (1.73%)

## perform enrichment analysis with n = 100 bootstraps

myTestRun <- calcEnrichment(myTestRun, n = 100)
#>
#> Bootstrapping...
#>
#> Match to pathway...
#> 35.43% of annotations were matched to pathway
#>
#> Perform enrichment analysis...

## example of the annotations, that now also include isobars

myTestRun$annotations[[
  sample(which(sapply(myTestRun$isobars_list, length) > 1), size = 1)]][1:10]
#>                                  isomer1
#>             "SM(d20:0/22:3(10Z,13Z,16Z))"
#>                                  isomer2
#>             "SM(d20:1(4E)/22:2(13Z,16Z))"
#>                                  isomer3
#>                          "SM(d18:2/24:1)"
#>                                  isomer4
```

```
#>                                  "SM(d42:3)"
#>                                      isobar1
#> "SM(d18:1(4E)/26:5(11Z,14Z,17Z,20Z,23Z))"
#>                                      isobar2
#>  "SM(d18:1(4E)/26:5(8Z,11Z,14Z,17Z,20Z))"
#>                                      isobar3
#>  "SM(d18:0/26:6(8Z,11Z,14Z,17Z,20Z,23Z))"
#>                                      isobar4
#> "SM(d14:0/30:6(12Z,15Z,18Z,21Z,24Z,27Z))"
#>                                      isobar5
#> "SM(d14:1(4E)/30:5(12Z,15Z,18Z,21Z,24Z))"
#>                                      isobar6
#> "SM(d14:1(4E)/30:5(15Z,18Z,21Z,24Z,27Z))"

## plot enrichment analysis, with q.values on x-axis, and with LION IDs

plotEnrichment(myTestRun, min.annotations = 5, q.value.cutoff = .05,
               by.statistic = "q.value")
```