FAMetA workflow

Maribel Alcoriza Balaguer

2021-11-18

FAMetA is a R package aimed to the estimation of FA import (I), de novo synthesis (S), fractional contribution of the 13C-tracers (D0, D1, D2), elongation (E) and desaturation (Des) based on mass isotopologue data.

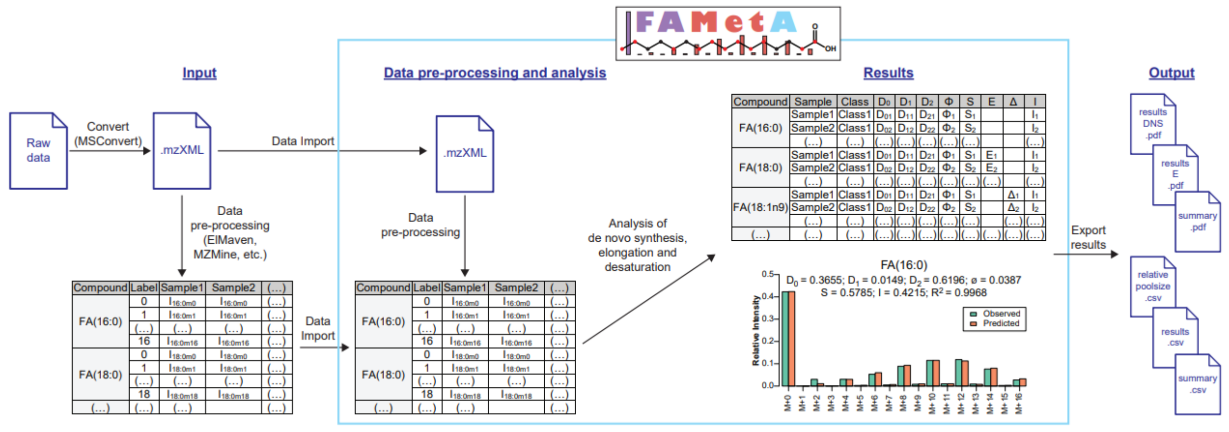

Figure 1: FAMetA abstract

Example files can be found at https://drive.google.com/drive/folders/1-mhqBd6W8VJkIYwVuIOr2Gn4Z9vBUN9-?usp=sharing.

Installation

```
# Install FAMetA
install.packages("FAMetA", dependencies = c("Depends", "Imports"))

# load library
library(FAMetA)
```

MS data preprocessing

Inputs

To start the MS analysis, raw data files need to be converted into mzXML format and a csv file with all the required information must be prepared with the following columns: sample (file names), acquisitionmode (MS in this particular case), sampletype (groups) and any other additional information such as protein levels for example, which can be used later to normalize your data.

Peak-picking, alignment and grouping using LipidMS package

Once all mzXML files have been obtained, peak picking have to be performed using the `batchdataProcessing()` function from LipidMS package. Alternatively, any other package or external software may be employed and data can be loaded from a csv file using `readfadatafile()` function.

```
library(FAMetA)
library(tools)

# Example files can be found at
# <https://drive.google.com/drive/folders/1-mhqBd6W8VJkIYwVuIOr2Gn4Z9vBUN9-?usp=sharing>.
# This dataset contains 12 samples of linfocites grown in culture media
# supplemented with 13Cglc, 13Cglc+iFASN, 13Cglc+iSCD and 13Cglc+iFADS2
# (3 replicates of each condition), 2 blank samples and 3 injections of a
# standards mix.

#==============================================================================#
# Data pre-processing using LipidMS package
#==============================================================================#

#=================#
# Read metadata

metadata <- read.csv("samples.csv", header = T, sep=",")

# check file names (they must include .mzXML)
if (!all(file_ext(metadata$sample) == "mzXML")){
  metadata$sample[file_ext(metadata$sample) != "mzXML"] <-
    paste(metadata$sample[file_ext(metadata$sample) != "mzXML"], ".mzXML", sep="")
}


#=================#
```

```r
# Set processing parameters

# Peak-picking
polarity <- "negative"
dmzagglom <- 15               # dmz and drt to generate bins/partitions for peak-picking
drtagglom <- 200              # max rt window for bins
drtclust <- 100               # drt window for clustering (redefines previous bins)
minpeak <- 8                  # min number of points to define a peak (MS1, MS2)
minint <- 100000              # at least minpeak points must have minint intensity
drtgap <- 5                   # max rt gap to fill missing points in a peak
drtminpeak <- 8               # min width of a peak when there are more than 1 peak in a EIC
drtmaxpeak <- 30              # max rt window for a EIC
recurs <- 10                  # max number of peaks in a EIC
sb <- 5                       # signal-to-baseline ratio (MS1, MS2)
sn <- 5                       # signal-to-noise ratio
weight <- 2                   # weight to assign new peaks
dmzIso <- 5                   # dmz for isotopes search
drtIso <- 5                   # drt for isotopes search

parallel <- TRUE              # parallel processing
ncores <- 4                   # number of cores


#==================#
# Peak-picking
msbatch <- batchdataProcessing(metadata = metadata,
                               polarity = polarity,
                               dmzagglom = dmzagglom,
                               drtagglom = drtagglom,
                               drtclust = drtclust,
                               minpeak = minpeak,
                               drtgap = drtgap,
                               drtminpeak = drtminpeak,
                               drtmaxpeak = drtmaxpeak,
                               recurs = recurs,
                               sb = sb,
                               sn = sn,
                               minint = minint,
                               weight = weight,
                               dmzIso = dmzIso,
                               drtIso = drtIso,
                               parallel = parallel,
                               ncores = ncores)
```

Once we have obtained the msbatch, we need to align samples and group all features to build the final feature matrix with `alignmsbatch()` and `groupmsbatch()` functions.

```r
#==================#
# Batch processing
dmzalign <- 10                # max dmz and rt to group peaks for alignment
drtalign <- 60                # max rt window for clustering in alignment
span <- 0.2                   # span for alignment
minsamplesfracalign <- 0.50   # min fraction of samples represented in a peak group to be used for alig
dmzgroup <- 10                # max dmz and rt to group peaks for grouping
```

```r
drtagglomgroup <- 50          # max rt window for clustering in grouping
drtgroup <- 10                # max rt difference within a peak group
minsamplesfracgroup <- 0.20   # min fraction of samples represented in a peak group to be kept


#==================#
# Alignment
msbatch <- alignmsbatch(msbatch, dmz = dmzalign, drt = drtalign, span = span,
                        minsamplesfrac = minsamplesfracalign,
                        parallel = parallel, ncores = ncores)

#==================#
# Grouping
msbatch <- groupmsbatch(msbatch, dmz = dmzgroup, drtagglom = drtagglomgroup,
                        drt = drtgroup, minsamplesfrac = minsamplesfracgroup,
                        parallel = parallel, ncores = ncores)


#==================#
# Save msbatch
save(msbatch, "msbatch.rda.gz", compress = TRUE)


# If any other external software is used for processing, data can be loaded from
# a csv file using the following function:
# fadata <- readfadatafile("externafadata.csv", sep=",", dec=".")

# In this case, go directly to data correction step.
```

Now we can annotate all fatty acids present in our samples.

FA annotation and manual curation

Next step consists in FA annotation. First, automatic FA annotation can be performed with `annotateFA()` function and identified FA can be checked using `plotFA()`.

```r
#==============================================================================#
# FA annotation
#==============================================================================#

#==================#
# Annotate FA
msbatch <- annotateFA(msbatch, dmz = 5)

#==================#
# plot peaks from identified FAs to check them
plots <- plotFA(msbatch, dmz = 10)

pdf("FAs.pdf")
for (p in 1:length(plots)){
  print(plots[[p]])
}
```

```
dev.off()

#=================#
# export annotations for curation
write.csv(msbatch$fas, file="faid.csv", row.names=FALSE)
```

Then, FA annotations can be modified at the csv file by removing rows of unwanted FA, modifying initial and end retention times or adding new rows with missing compounds. Here, unique compound names with the nomenclature "FA(16:1)n7", where n7 (omega-7) indicates the last double bound position, are required to differentiate between FA isomers. In case of unknown positions, x, y and z letters are allowed (i.e. FA(16:1)nx). In addition, internal standards for later normalization can also be added at this point in a new row indicating IS in the compound name column. Check faid.csv and faid_curated.csv files to see an example. Changes can also be performed directly on the msbatch using addFA, removeFA, changeFArt and searchIS functions. See documentation for details.

```
#==============================================================================#
# FA curation
#==============================================================================#

#=================#
# read csv file with modified annotations
faid <- read.csv("faid_curated.csv", sep=",", dec=".")

#=================#
# change FA annotations
msbatch <- curateFAannotations(msbatch, faid)

#=================#
# plot FA peaks again to check identities
plots <- plotFA(msbatch, dmz = 10)

pdf("FAs_curated.pdf")
for (p in 1:length(plots)){
  print(plots[[p]])
}
dev.off()
```

Search isotopes

Once all FA have been correctly identified and defined at the msbatch, isotopes have to be searched based on peak limits (iniRT and endRT), mass tolerance (in ppm) and peak correlation.

```
#==============================================================================#
# Search FA isotopes and get the fadata object
#==============================================================================#
fadata <- searchFAisotopes(msbatch, dmz = 10, coelCutoff = 0.6)


# if you want to save fadata in a csv to subset it for example:
df <- cbind(rbind(fadata$fattyacids, data.frame(Compound="IS", Label="")),
            rbind(fadata$intensities, fadata$IS))
df <- rbind(c("", "sampletype", fadata$metadata$sampletype),
            c("Compound", "Label", colnames(fadata$intensities)), df)
write.table(df, file="fadata.csv", sep=",", col.names = FALSE, row.names = FALSE)

# and then, you could read it again using:
fadata <- readfadatafile("fadata.csv", sep=",", dec=".")
```

`searchFAisotopes()` function will return a simplified fadata list which will be used for the metabolic analysis. In case external software have been used to process data, `readfadatafile()` function can import a csv file and return a similar fadata list. An example data file can be found at https://drive.google.com/drive/folders/1-mhqBd6W8VJkIYwVuIOr2Gn4Z9vBUN9-?usp=sharing.

```
#==============================================================================#
# Import FA data
#==============================================================================#
inputfile <- "externalfadata.csv"
fadata <- readfadatafile(inputfile, sep=",", dec=".")
```

Data correction

Previous to metabolic analysis, data must be corrected and normalized using `dataCorrection()` function, which will execute four steps: data correction for natural abundance of 13C using the accucor algorithm (Su et al., 2017), data normalization with internal standards (optional), blank substraction (optional) and external normalization (applied if additional columns are added to the metadata file).

In case blank substraction have to be applied, blankgroup argument must contain the group name used in metadata to define blank samples. Similarly, for esternal normalization, externalnormalization argument must contain the column name at the metadata file that contains the values to be used.

```
#==============================================================================#
# Data correction
#==============================================================================#
fadata <- dataCorrection(fadata, blankgroup = "Blank")

# Alternatively, to add external normalization:
# fadata <- dataCorrection(fadata, blankgroup = "blank",
# externalnormalization = "protein")
```

Metabolic analysis

Finally, metabolic analysis can be performed. To this end, 3 different functions have to be run sequentially: `synthesisAnalysis()`, `elongationAnalysis()`and `desaturationAnalysis()`. Then results will be summarize and represented using `summarizeResults()`.

Synthesis analysis

Synthesis analysis will model FA data for FA up to 16 carbons to estimate 13C-tracer contribution to the acetyl-CoA pool for FA synthesis (D) and the FA fraction that has been synthesized de novo. D0, D1 and D2 represent the contribution of M+0, M+1 and M+2 acetate, respectively, and P (phi) is the overdispersion parameter of the quasi-multinomial distribution.

```r
#==============================================================================#
# Metabolic analysis
#==============================================================================#

#=================#
# Synthesis analysis
fadata <- synthesisAnalysis(fadata=fadata, R2Thr = 0.95, maxiter = 1e3,
                            maxconvergence = 100, startpoints = 5)

# If inhibitors have been used, make sure D2 has not been underestimated. If so,
# D2 could be set as the one calculated for 13-Glc Control samples to improve
# the results:

# D2 <- fadata$synthesis$results$D2[fadata$synthesis$results$FA == "FA(16:0)"]
# fadata$synthesis$results$Group[fadata$synthesis$results$FA == "FA(16:0)"]

# D2[4:12] <- rep(mean(D2[1:3]))

# relaunch synthesis analysis fixing D2
# fadata <- synthesisAnalysis(fadata=fadata, R2Thr = 0.95, maxiter = 1e3,
#                              maxconvergence = 100, startpoints = 5, D2 = D2)


# Explore results
View(fadata$synthesis$results)
View(fadata$synthesis$predictedValues)
pdf("plotsSynthesis.pdf")
for (f in 1:length(fadata$synthesis$plots)){
  for (s in 1:length(fadata$synthesis$plots[[f]])){
    print(fadata$synthesis$plots[[f]][[s]])
  }
}
dev.off()

# to use multinomial distribution without over dispersion, set P parameter to 0
fadata <- synthesisAnalysis(fadata=fadata, R2Thr = 0.95, maxiter = 1e3,
                             maxconvergence = 100, startpoints = 5, P = 0)
```

D0, D1, D2 can also be fixed if they are known. This is particularly useful in case inhibitors have been used as they could reduce S below the confidence interval and thus, S and D parameters could be misestimated.

Elongation analysis

Main route of de novo synthesis plus elongation starts at 16 carbons and then adds blocks of 2 carbons. Therefore, isotopologue distributions for FA longer than 16 carbons will be modeled taking into account de novo synthesis until FA(16:0), followed by single and independent elongation steps (E1, E2 . . . , En). Parameters D0, D1 and D2 are imported from FA(16:0)/FA(14:0) and thus the only relevant parameters to be estimated in the elongation analysis are Ei and I.

For n6 and n3 series, elongation is expected from FA(18:2)n6 and FA(18:3)n3 so that synthesis (S16:0) and first elongation step (E1) are set to 0.

```r
#==================#
# Elongation analysis
fadata <- elongationAnalysis(fadata, R2Thr = 0.95, maxiter = 1e4,
                             maxconvergence=100, startpoints = 5, DThr = 0.1)


# Explore results
View(fadata$elongation$results)
View(fadata$elongation$predictedValues)
pdf("plotsElongation.pdf")
for (f in 1:length(fadata$elongation$plots)){
  for (s in 1:length(fadata$elongation$plots[[f]])){
    print(fadata$elongation$plots[[f]][[s]])
  }
}
dev.off()
```

Desaturation analysis

Finally, after synthesis and elongation analysis have been run, desaturation analysis can be performed.

```r
#==================#
# Desaturation analysis
fadata <- desaturationAnalysis(fadata)

# Explore results
View(fadata$desaturations$results)
```

Results and graphical outputs

To ease data interpretation, `summarizeResults()` function has been been designed to generate different results tables and heatmaps.

```r
#==================#
# Summarize results
fadata <- summarizeResults(fadata, controlgroup = "Control13Cglc")

#==================#
# Save fadata
save(fadata, file="fadata.rda")



#==================#
# Export results
write.csv(fadata$results$results, file = "results.csv", row.names=FALSE)
write.csv(fadata$results$summary, file = "summary.csv")
write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$mid),
                  fadata$mid),
            file = "mid.csv", sep=",", col.names = FALSE)
```

```r
write.table(rbind(fadata$metadata$sampletype,
                    colnames(fadata$synthesis$predictedValues),
                    fadata$synthesis$predictedValues),
              file = "predictedmid.csv", sep=",", col.names = FALSE)


pdf("relativepoolsizeRaw.pdf")
print(fadata$results$heatmaps$relativepoolsize$raw$plot)
dev.off()

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$heatmaps$relativepoolsize$raw$values),
                  fadata$results$heatmaps$relativepoolsize$raw$values),
            file = "relativepoolsizeRaw.csv", sep=",", col.names = FALSE)


pdf("relativepoolsizeZscore.pdf")
print(fadata$results$heatmaps$relativepoolsize$zscore$plot)
dev.off()

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$heatmaps$relativepoolsize$raw$values),
                  fadata$results$heatmaps$relativepoolsize$zscore$values),
            file = "relativepoolsizeZscore.csv", sep=",", col.names = FALSE)

if ("log2FC" %in% names(fadata$results$heatmaps$relativepoolsize)){
  pdf("relativepoolsizeLog2FC.pdf")
  print(fadata$results$heatmaps$relativepoolsize$log2FC$plot)
  dev.off()

  write.table(rbind(fadata$metadata$sampletype,
                    colnames(fadata$results$heatmaps$relativepoolsize$log2FC$values),
                    fadata$results$heatmaps$relativepoolsize$log2FC$values),
              file = "relativepoolsizeLog2FC.csv", sep=",", col.names = FALSE)
}

pdf("resultsRaw_endogenouslySynthesized.pdf")
print(fadata$results$heatmaps$synthesized$raw$plot)
dev.off()

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$heatmaps$synthesized$raw$values),
                  fadata$results$heatmaps$synthesized$raw$values),
            file = "resultsRaw_endogenouslySynthesized.csv", sep=",",
            col.names = FALSE)


if ("log2FC" %in% names(fadata$results$heatmaps$synthesized)){
  pdf("resultsLog2FC_endogenouslySynthesized.pdf")
  print(fadata$results$heatmaps$synthesized$log2FC$plot)
  dev.off()

  write.table(rbind(fadata$metadata$sampletype,
                    colnames(fadata$results$heatmaps$synthesized$log2FC$values),
                    fadata$results$heatmaps$synthesized$log2FC$values),
              file = "resultsLog2FC_endogenouslySynthesized.csv", sep=",",
              col.names = FALSE)
}


#=================#
# Isotopologue distributions: observed vs predicted

pdf("isotopologueDistributions.pdf")
for (f in 1:length(fadata$synthesis$plots)){
  for (s in 1:length(fadata$synthesis$plots[[f]])){
    print(fadata$synthesis$plots[[f]][[s]])
  }
}
for (f in 1:length(fadata$elongation$plots)){
    for (s in 1:length(fadata$elongation$plots[[f]])){
      print(fadata$elongation$plots[[f]][[s]])
    }
  }
dev.off()



#=================#
# Reorganized tables for synthesis and elongation parameters (S16, E1, E2, E3,
# E4 and E5)

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$S16),
                  fadata$results$allparameters$S16),
            file = "S16.csv", sep=",", col.names = FALSE)

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$E1),
                  fadata$results$allparameters$E1),
            file = "E1.csv", sep=",", col.names = FALSE)

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$E2),
                  fadata$results$allparameters$E2),
            file = "E2.csv", sep=",", col.names = FALSE)

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$E3),
                  fadata$results$allparameters$E3),
            file = "E3.csv", sep=",", col.names = FALSE)

write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$E4),
                  fadata$results$allparameters$E4),
            file = "E4.csv", sep=",", col.names = FALSE)
```

```r
write.table(rbind(fadata$metadata$sampletype,
                  colnames(fadata$results$allparameters$E5),
                  fadata$results$allparameters$E5),
            file = "E5.csv", sep=",", col.names = FALSE)
```

Customizable fatty acids parameters

In case any additional fatty acid needs to be added (i.e. extra unknown fatty acids such as FA(18:2)nv) or some parameters changed (i.e. add E1 step for FA(18:3)n3 or n6), it is possible to change the parameters database from FAMetA.

```r
#==================#
# Customize parameters database
parameters <- FAMetA::parameters

# Add a new unknown FA(18:1)
newrow <- data.frame(FattyAcid = "FA(18:1)nv",
                     M = 18,
                     S16 = 1,
                     E1 = 1,
                     E2 = 0,
                     E3 = 0,
                     E4 = 0,
                     E5 = 0)


parameters <- data.frame(rbind(parameters, newrow))
parameters <- parameters[order(parameters$FattyAcid),]
View(parameters)

# Change fatty acid settings: add E1 step for FA(18:3)n6
parameters$E1[parameters$FattyAcid == "FA(18:3)n6"] <- 1


# Then add the parameters argument to elongationAnalysis and summarizeResults
# functions
fadata <- elongationAnalysis(fadata, R2Thr = 0.95, maxiter = 1e4,
                             maxconvergence=100, startpoints = 5, D2Thr = 0.1,
                             parameters = parameters)

fadata <- FAMetA:::summarizeResults(fadata, controlgroup = "H460_13Cglc",
                                    parameters = parameters)
```

This same strategy can be used to modify the desaturations database (desaturationdb).

If you have any further questions, please do not hesitate to contact us at maribel_alcoriza@iislafe.es.