compMS2Miner_Workflow

WMB Edmands

February 24, 2017

Matches MS1 features to MS2 spectra (.mzXML/.mzML/.mgf) files based on a mass-to-charge and retention time tolerance. Adducts and in-source fragments identified in the MS1 data set (e.g. using the CAMERA package) can be used to guide the annotation process simply adding a 4th column containing the adducts identified. Composite spectra and other supporting data can subsequently be visualized during any stage of the compMS2Miner processing workflow. Composite spectra can be denoised, ion signals grouped and summed, substructure groups identified, common Phase II metabolites predicted, both correlation and spectral similarity networks calculated, mean maximum first network neighbour chemical similarity calculation, molecular descriptor and random forest based retention time prediction and features matched to data bases monoisotopic mass data, spectral databases in the form of a NIST .msp ASCII file (examples given) and *in silico* MS2 fragmentation data using both the latest command line versions of MetFrag and CFM.

compMS2Miner offers the option of an entirely autonomous first-pass annotation identification workflow using a build consensus score function which takes into account multiple lines of evidence in support of an identification and automatically identifies any annotation above a minimum mean consensus score. Additionally, the contribution of each of these consensus score to a weighted mean consensus score can be globally optimized using a differential evolution algorithm.

The resulting fully analysed and annotated data can then be readily curated by publishing as online shiny application on shinyapps.io, as a zip file which can be shared or by sending to a local or online couchDB database. compMS2Miner makes the establishment of a stand-alone database for any metabolomics experiment extremely straightforward.

At any stage during the compMS2 workflow a compMS2 class object can be visualized with a Shiny web-ready application *compMS2Explorer*. *Composite MS2 Explorer* can be thought of as a study specific database for your untargetted metabolomics analysis dataset. Full usage of the *compMS2Explorer* application functionality requires an internet connection. The end result of following the workflow within this document using the example data provided can be visualized using the function, comments made in the metID comments tab can be saved on closing the application by assigning the compMS2Explorer output to an object:

```r
library(compMS2Miner)
# assign any metabolite identification comments to a new or the same "compMS2" object
compMS2Example_commented <- compMS2Explorer(compMS2Example)
```

An Rmarkdown (.Rmd) file pdf skeleton is provided within the package. This file has a typical workflow using xcms, CAMERA and compMS2Miner and details each of the compMS2Miner functions in sequence. This skeleton can be used to create a reproducible workflow of your peak-picking and MS^2 data annotation and the resulting pdf file can be bundled alongside your published application.

```r
rmarkdown::draft("metabolomic_workflow_example.Rmd", template = "compMS2Template",
                 package = "compMS2Miner", create_dir = TRUE)
```

The following example illustrates the compMS2Miner workflow:

1. construct compMS2 class object.

From MS2 data (in the .mzXML file format) and an MS1feature table. The compMS2 object can also be constructed in as a parallel computation in the case of large peak tables and/ or larger numbers of MS2 data files.

```r
# file path example MS1features in comma delimited csv file
# (see ?example_mzXML_MS1features for details).
MS1features_example <- system.file("extdata", "MS1features_example.csv",
                                    package = "compMS2Miner")
# mzXml file examples directory (can also be .mzML or .mgf files)
mzXmlDir_example <- dirname(MS1features_example)
# observation MS1 feature table column names character vector for corrNetwork function
obsNames <- c(paste0(rep("ACN_80_", 6), rep(LETTERS[1:3], each=2), rep(1:2, 3)),
              paste0(rep("MeOH_80_", 6), rep(LETTERS[1:3], each=2), rep(1:2, 3)))
# use parallel package to detect number of cores
nCores <- parallel::detectCores()

# read in example peakTable
peakTable <- read.csv(MS1features_example, header=TRUE, stringsAsFactors=FALSE)
# create compMS2 object
compMS2Demo <- compMS2Construct(MS1features = peakTable,
                                msDataDir = mzXmlDir_example, nCores=nCores,
                                mode = "pos", precursorPpm = 10, ret = 20,
                                TICfilter = 10000)

# View summary of compMS2 class object at any time
compMS2Demo
```

2. Dynamic noise filtration.

filter variable noise from the data using a dynamic noise filter.

```r
# dynamic noise filter
compMS2Demo <- deconvNoise(compMS2Demo, "DNF")
# View summary of compMS2 class object at any time
compMS2Demo
```

3. Intra-spectrum ion grouping and inter-MS2 file spectra grouping with signal summing.

group and sum ions from different scans and then combine summed ion composite spectra across multiple files. This create a single composite spectra for each MS1 EIC matched to MS2 precursor scans. Optionally, a minimimum inter-spectrum similarity filter (dot product) can be applied where only spectra with a minimum spectral similarity (see argument specSimFilter in ?combineMS2.Spectra) will be combined in the resultant composite spectrum. However, if none of the spectra of a group have any similarity to one another the spectrum with the highest mean precursor intensity will be returned.

Additionally potential contaminants can be detected and removed from downstream analysis. Contaminants in this context are defined as repeated sequences of isobaric precursor masses of sufficient minimum length and high spectral similarity (dot product) seperated by a maximum retention time gap. These type of background ion fragmentation events can however be avoided during data-dependent MS/MS acquisition by addition to an exclusion list or using dynamic exclusion. However without foreknowledge of these isobaric contaminants, in many cases whole metabolome identification efforts can often contain many of these unwanted spectra.

```r
# intra-spectrum ion grouping and signal summing
compMS2Demo <- combineMS2(compMS2Demo, "Ions")
# View summary of compMS2 class object at any time
compMS2Demo
```

```
#inter spectrum ion grouping and signal summing if the argument specSimFilter
# is supplied (values 0-1) then any spectrum below this spectral similarity
# threshold (dot product) will not be included in the composite spectrum generated.
compMS2Demo <- combineMS2(compMS2Demo, "Spectra", specSimFilter=0.8)
# View summary of compMS2 class object at any time
compMS2Demo
# remove any potential contaminants based on repeated isobaric precursor masses spaced by a maximum
# retention time gap and of high spectral similarity
compMS2Demo <- combineMS2(compMS2Demo, 'removeContam', maxRtGap=Inf, nContams=4)
```

4. Possible substructure identification.

Characteristic neutral losses/ fragments of electrospray adducts and metabolites from literature sources (?Substructure_masses for details). In addition the substructures detected with a minimum summed relative intensity above the argument minSumRelInt (default = 70%) will be automatically added to the Comments table in the compound_class column and flagged as subStructure.prob (e.g. sulfate (subStructure.prob)). In this way the comments table can be automatically filled but can be modified if necessary.

```
# annotate substructures
compMS2Demo <- subStructure(compMS2Demo, "Annotate")
# identify most probable substructure annotation based on total relative intensity
# explained
compMS2Demo <- subStructure(compMS2Demo, "prob")
# summary of most probable substructure annotation
mostProbSubStr <- subStructure(compMS2Demo, "probSummary")
```

5. Metabolite identification methods.

A variety of metabolite identification methods are implemented through the function metID (see ?metID) for further details.

Annotate features using HMDB (default)

```
# annotate any phase II metabolites detected
subStrMassShift <- c(42.010565, 119.004101, 176.03209, 255.988909,
                     305.068159, 57.021464, 161.014666, 79.956817)
names(subStrMassShift) <- c("acetyl", "cysteine", "glucuronide",
                            "glucuronide sulfate", "glutathione", "glycine",
                            "mercapturate", "sulfate")
subStrTypesDetected <- sapply(names(subStrMassShift), function(x){
                              grepl(x, Comments(compMS2Demo)$compound_class,
                                    ignore.case=TRUE)})
subStrMassShift <- subStrMassShift[subStrTypesDetected]
#  common mandatory esi adducts i.e. added to those detected in the MS1 data by CAMERA
mandEsiAdducts <- c('[M-H2O-H]-', '[M+Na-2H]-', '[M+Cl]-', '[M+K-2H]-', '[M-H+HCOOH]-',
                    "[M-H+CH3COOH+Na]-", "[M-H+CH3COOH]-")
# what if any substructure detected?
subStrMassShift <- subStrMassShift[names(subStrMassShift) %in% names(mostProbSubStr)]
#  common mandatory esi adducts i.e. added to those detected in the MS1 data by CAMERA
#  adducts supplied as structures can be automatically interpreted with the adduct2mass function
#  N.B. The default is to use a large number of adduct and fragments from the
```

```r
#  paper of Stanstrup et. al.. See ?metID.dbAnnotate for details
mandEsiAdducts <- c('[M+H]+', '[M+NH4]+', '[M+Na]+','[M+CH3OH+H]+', '[M+CH3COONa]+',
                    '[M+K]+')
# elements to consider in the database annotation. Any structure containing an
# element symbol not in this list will be filtered out
includeElements=c('C', 'H', 'N', 'O', 'P', 'S', 'F', 'Cl', 'Br', 'I')
# annotate composite MS2 matched MS1 features to metabolomic databases (default
#is HMDB, also LMSD (lipidMaps), DrugBank, T3DB and ReSpect databases can also be queried).
#Warning: this may take 2-3 mins as large number of query masses
compMS2Demo <- metID(compMS2Demo, "dbAnnotate", esiAdduct=mandEsiAdducts,
                     includeElements=includeElements)
```

Annotate features using LMSD (LipidMaps Structural Database)

```r
compMS2Demo <- metID(compMS2Demo, "dbAnnotate", metDB=LMSD, esiAdduct=mandEsiAdducts,
                     includeElements=includeElements)
```

Annotate features using Drugbank

```r
compMS2Demo <- metID(compMS2Demo, "dbAnnotate", metDB=drugBank, esiAdduct=mandEsiAdducts,
                     includeElements=includeElements)
```

Annotate features using T3DB

```r
compMS2Demo <- metID(compMS2Demo, "dbAnnotate", metDB=T3DB, esiAdduct=mandEsiAdducts,
                     includeElements=includeElements)
```

Annotate features using ReSpect database

```r
compMS2Demo <- metID(compMS2Demo, "dbAnnotate", metDB=ReSpect, esiAdduct=mandEsiAdducts,
                     includeElements=includeElements)
```

match composite spectra to spectral databases as .msp files

```r
# match composite spectra to spectral databases as .msp files
compMS2Demo <- metID(compMS2Demo, 'matchSpectralDB')
```

match to lipidBlast with SMILES added on GitHub repository

N.B. It is faster to utilize local copies of msp files rather than reading from the internet. If you are anticipating frequent access of msp files then create a local archive by downloading from MoNa or cloning the msp file repository on GitHub (with SMILES codes added by OpenBabel). https://github.com/WMBEdmands/mspFiles. In the case that smiles codes are not contained in the msp file then if a copy of the openBabel (openbabel.org) command line tool is installed in your path then the matchSpectralDB function will automatically convert InChI codes to smiles.

```
lbMspFiles <- paste0('https://raw.githubusercontent.com/WMBEdmands/mspFiles/master/MoNA-export-Librarie
for(i in 1:5){
 compMS2Demo <- metID(compMS2Demo, 'matchSpectralDB', mspFile=lbMspFiles[i])
}
```

select most probable annotations based on substructures detected

A substructure score is calculated based on the prescence of susbstructures detected in each SMILES code. This score can be used to rank metabolites based on the substructures detected. The argument minTimesId (default=2) is set to 1 this means that a substructure type (e.g. sulfate, phosphatidylcholine) must be detected at least this many times to be considered for the similarity scoring. If a substructure has been identified once by chance then these will not be considered if this argument is set to 2. For phosphatidylcholines/sphingomyelins for instance often only 1 fragment is detected (the PC head group ~184 m/z).

```
# select most probable annotations based on substructures detected
compMS2Demo <- metID(compMS2Demo, "dbProb", minTimesId=1)
```

Phase II metabolite prediction from SMILES codes

Crude single Phase II metabolite structural prediction. If sulfate, glucuronide or glycine conjugates have been annotated then approriate functional groups are identified in each SMILES code (i.e. hydroxyl, amine and carboxylic acid groups) and then the first appropriate function group in the SMILES code is used to generate a possible Phase II metabolite structure. These structures then replace the previous SMILES code in the SMILES column. In addition the Phase II metabolite type is added to the DBid column in the Best Annotations and the substructure type in the SubStr_type column removed to indicate that this function has been run. You can then visualize the structure in compMS2Explorer by clicking and viewing using the PUG-rest service of pubMed.

```
# predict Phase II metabolites from SMILES codes
compMS2Demo <- metID(compMS2Demo, "predSMILES")
```

In silico fragmentation prediction using the command line version of MetFrag CL 2.3

The adduct types considered are customizable see the default table data(metFragAdducts). N.B. These adduct names must perfectly match the adduct names supplied to metID.dbAnnotate and the internal adduct2mass function.

```
# metFrag in silico fragmentation.
compMS2Demo <- metID(compMS2Demo, "metFrag")
```

In silico fragmentation prediction using the command line version of CFM fraggraph-gen

This function does not use the whole functionality of CFM and merely generates all possible fragments using the fraggraph-gen tool. This only considers the (de-)protonated annotations (e.g. [M+H]+ or [M-H]-) and no other ESI adduct types. This commandline tool was determined to be significantly faster and more practical for large numbers of annotations than other CFM command line tools identified. It must be noted that improved results but with additional computational time constraints could be obtained by using other trained CFM tools or the website interface. A link is included in the compMS2Explorer app to both the MetFrag and CFM online interfaces.

```
# CFM in silico fragmentation.
compMS2Demo <- metID(compMS2Demo, "CFM")
```

spectral similarity network, MetMSLine data-preprocessing and correlation

network calculation. Both a spectral similarity network and correlation network can be calculated and visualized within the compMS2Explorer interface. Spectral similarity (dot product) is calculated for both the fragment pattern and also the precursor - fragment neutral loss pattern. In this way spectra with different fragments but shared neutral loss pattern will be connected. Whether the spectra are connected by highly similar fragmentation pattern or shared neutral loss pattern is highlighted on the spectral similarity plot. In this way clusters of similar spectra can be rapidly identified originating from very similar compounds. Correlation analysis provides a means to identify shared biological origins/pathways or related metabolites in the dataset. Features without MS/MS spectra can also be included in this network. In future versions annotation of these features without MS/MS spectra will be made possible.

```
# calculate spectral similarity network (dot product >= 0.8 default)
compMS2Demo <- metID(compMS2Demo, 'specSimNetwork')
##############################################################
### data pre-processing MS1features with MetMSLine package ###
##############################################################

if(!require(MetMSLine)){
  # if not installed then install from github
  devtools::install_github('WMBEdmands/MetMSLine')
  require(MetMSLine)
}

# zero fill
peakTable <- zeroFill(peakTable, obsNames)
# log transform
peakTable <- logTrans(peakTable, obsNames)

########################################################
######### end MetMSLine pre-processing #################
########################################################

# add correlation network using pre-processed MS2 matched peak table
compMS2Demo <- metID(compMS2Demo, method='corrNetwork', peakTable, obsNames,
                     corrMethod='pearson', corrThresh=0.90, MTC='none', MS2only=3)
```

mean maximum first correlation/spectral similarity network neighbour chemical

similarity scoring and optional first-pass automatic possible annotation identification. This annotation is based on a mimimum mean maximum tanimoto chemical similarity with all of its neighbours (considering both correlation and spectral similarity neighbours). That is annotations for each feature are ranked by the weighted mean of the highest chemical similarities to any of it's connected feature. In this way clusters of highly related compounds can be automatically annotated and the names added directly to the Comments table. Any features already matched to a database that is annotated by the metID.matchSpectralDB function will not be overwritten with this function.

```
# conduct mean maximum first network neighbour chemical similarity ranking
compMS2Demo <- metID(compMS2Demo, 'chemSim', autoPossId=TRUE)
```

Molecular descriptor and Random Forest based retention time prediction modelling.

This function uses any metabolite present in either a user supplied table of standards see argument standardsTable for details or any spectra already automatically or manually annotated in the Comments table. To see the Comments table at any time used the function Comments(object). The function uses pre-calculated molecular descriptors for all internal databases to save time. The molecular descriptors were calculated using the rcdk package. Any structures without molecular descriptors will be calculated automatically (e.g. predicted Phase II metabolites or external databases not contained in the package).

Then molecular descriptors are filtered to remove those will high numbers of missing values, zeros, non-zero variance and high inter-correlation. Then the caret package is used to identify the optimum set of molecular descriptors by recursive feature elimination (RFE). The final model is then used to calculate the predicted retention times of all annotations for each composite spectrum/feature and the degree of deviation from the expected retention time. This can be then used to rank annotations based on expected retention time. The final model can be accessed using the function rtPred(object) and then plotted.

```r
# conduct retention prediction modelling
compMS2Demo <- metID(compMS2Demo, 'rtPred')
plot(rtPred(compMS2Demo)$rfModel)
```

Build metabolite annotation consensus using up to 8 parameters for identification and automatic possible identity annotation.

see ?metID.buildConsensus for details of the 8 options included by default (massAccuracy, spectralDB, inSilico, substructure, rtPred, chemSim, pubMed, corrMolDesc) A simple arithmetic mean of any combination of the annotation ranking scores can be used to build a consensus rank score of annotations. The most highly ranked annotations above a minimum mean consensus score can also be automatically added to the Comments table and flagged as metID.buildConsensus in the user_comments column. The include option 'pubMed' will return the number of references from the pubMed database for that compound name. Common phospholipid abbreviations will be also recognized before this search. This option can only be performed during specific hours in the evenings and on weekends to prevent overload of the PubMed servers. The function will pause and wait for the correct time before commencing but it is necessarily a slow process. See metID.buildConsensus for details.

```r
# build consensus metabolite annotation see ?metID.buildConsensus for details
compMS2Demo <- metID(compMS2Demo, 'buildConsensus', autoPossId=TRUE,
                     include=c('massAccuracy', 'inSilico', 'rtPred', 'chemSim',
                               'substructure'))
```

Differential Evolution based global optimization of weights of any combination of 8 parameters selected for consensus annotation identification and automatic possible identity annotation.

The differential evolution search heuristic can be used to globally optimize the weight of each annotation score to a weighted mean consensus score. The strategy seeks to minimize the overall mean rank of the "correct" annotations (i.e. those in the Comments table possible_identity column) which were either automatically or manually added by the user. A plot of progress is updated in the console every 20 population members (default) and the search algorithm continues up to 100 generations (default) however convergence may occur earlier and the maximum population number can therefore be reduced.

```r
# use a differential evolution genetic algorithm to identify the optimum weights
# of each of the 6 buildConsensus parameters ()
compMS2Demo <- metID(compMS2Demo, 'optimConsensus', autoPossId=TRUE,
                     include=c('massAccuracy', 'inSilico', 'rtPred', 'chemSim',
                               'substructure'), itermax=40)
```

6. Optional curate data as zip file, in CouchDB and/or publish results as a shiny application.

compMS2 class objects can be sent to either a local or online couchDB database. Furthermore, the results of the compMS2Miner workflow can be published on the web as a shiny application to share metabolite identification information or as a self-contained zip file using the publishApp function. Spectral hash identifiers (Splash codes) are automatically generated for every spectrum on publication/zip file/msp file generation.

```r
# publish your app to shinyapps.io see ?publishApp for more details
# you may need to install the rsconnect and shinyapps packages and also sign up for a shinyapps.io acco
# quick guide here for setting up your account: http://shiny.rstudio.com/articles/shinyapps.html
publishApp(compMS2Demo, appName='compMS2Demo',
           addFiles=system.file("doc", "compMS2Miner_Workflow.pdf",
                                package = "compMS2Miner"))
```

Or the app can be published as a self-contained zip file (which can be uploaded to an online repository), shared with collaborators as a standalone file. All of the files necessary to run the application are contained in the zip file bundle. This means that even in the future the application will be viewable. You can add also optionally add your R markdown pdf file and other additional files to the zip file which will then be displayed in the application, this allows the code used to generate the app to be included along with the application.

```r
publishApp(compMS2Demo, appName='compMS2Demo',
           writeDir='C:/',
           addFiles=system.file("doc", "compMS2Miner_Workflow.pdf",
                                package = "compMS2Miner"))
```

Alternatively absolutely everything can be published together so anyone can reproduce the workflow this could include the peak-picker output table and the mzXML files also. The zip file is compressed but including the mzXML files and peak table will make the file much larger. However with this approach all of the data needed to fully reproduce the results is included in one bundle.

```r
mzXmlFiles <- list.files(mzXmlDir_example, full.names = TRUE, pattern='\\.mzXML$')
publishApp(compMS2Demo, appName='compMS2Demo',
           writeDir='C:/',
           addFiles=c(system.file("doc", "compMS2Miner_Workflow.pdf",
                                  package = "compMS2Miner"),
                      system.file("extdata", "MS1features_example.csv",
                                  package = "compMS2Miner"), mzXmlFiles))
```

Post "publication" the zip file can be directly opened using compMS2Explorer, modified and potentially re-published

```r
# full path to zip file written by publishApp function
compMS2Demo <- compMS2Explorer('C:/compMS2Demo/compMS2Demo.zip')
```

An msp database text file can be generated directly from the compMS2 class object

this allows an in-house database to be constructed. These files can then be rapidly matched to new datasets using the metID.matchSpectralDB function. In this way previously annotated spectra can be matched limiting redundancy in the investigator's efforts.

```r
metID(compMS2Demo, 'compMS2toMsp')
```

sessionInfo

```r
sessionInfo()
```

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 14393)
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] compMS2Miner_2.2.3 knitr_1.15.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.9        lattice_0.20-34    digest_0.6.12
##  [4] rprojroot_1.2      MASS_7.3-45        grid_3.3.2
##  [7] MatrixModels_0.4-1 nlme_3.1-131       backports_1.0.5
## [10] stats4_3.3.2       magrittr_1.5       evaluate_0.10
## [13] stringi_1.1.2      minqa_1.2.4        nloptr_1.0.4
## [16] Matrix_1.2-8       rmarkdown_1.3      splines_3.3.2
## [19] lme4_1.1-12        tools_3.3.2        stringr_1.1.0
## [22] yaml_2.1.14        colorspace_1.3-2   htmltools_0.3.5
```