PODIUM USER MANUAL

Table of Content

Introduction

This document contains instructions on how to install and run PODIUM, as well as a description of the settings and logic that underlie its peak pairing algorithm. A driver script that will automatically perform the entire analysis given a set of user-defined parameters is also available on the PODIUM GitHub repository[1]. Instructions on how to set and run this driver script are also included below. Further details regarding PODIUM can be found in the PODIUM paper[2]. Finally, as this package is an extension of XCMS version 1.39.6, for more information on using the standard XCMS methods, we refer the reader to the XCMS package R vignette[3] and the original XCMS papers (Smith et al., 2006, Tautenhahn et al., 2008).

Installation

Use of PODIUM requires an installation of R 3.0.2, R tools version 31 (if on Windows), and two packages, PODIUM and PODIUMmethods, along with their dependencies. The following instructions are for a Windows-based installation, modifications for installing on other platforms are provided at the end. We recommend running R with administrative privileges during installation as it allows for the updating of R's base packages, which is recommended but not required.

1. Download and install R 3.0.2 from CRAN
    - https://cloud.r-project.org/bin/windows/base/old/3.0.2

2. Download and install R tools version 31 from CRAN
    - https://cran.r-project.org/bin/windows/Rtools/Rtools31.exe
    - Make sure to choose to add R tools to your PATH during install
    - R tools is necessary to install packages from source that contain non-R code, such as the PODIUM package

3. Install XCMS and multtest via Bioconductor
    - Note: This step is optional but is the easiest way to automatically install all of the necessary dependencies for the PODIUM package (as they are the same as those of the XCMS package)
    - Run R 3.0.2 and enter the following commands (>)
    - Set R to use CRAN's archive repository for installing packages
        - > options(repos=c(CRAN="https://cran-archive.r-project.org/"))
    - Download and install XCMS
        - > source('http://www.bioconductor.org/biocLite.R')
        - > biocLite("xcms")
        - If prompted to update packages, select yes (may require administrative privileges)

[1] https://github.com/chapple-lab/podium
[2] https://doi.org/10.1093/plcell/koaa046
[3] https://bioconductor.org/packages/2.13/bioc/html/xcms.html

- Download and install multtest
  - > biocLite("multtest")
- Update all packages to the latest available version from the CRAN archive
  - > update.packages()
  - Say yes to all prompts
  - (May require administrative privileges for some packages)

4. Download and install PODIUM and PODIUMmethods from GitHub
   - Download the latest releases for PODIUM and PODIUMmethods
     - PODIUM: https://github.com/chapple-lab/podium/releases
     - PODIUMmethods: https://github.com/chapple-lab/podiumMethods/releases
     - Note: Please download the package archive (the .tar.gz file) and not the source code
   - Run R 3.0.2 and enter the following commands (>)
   - Set R to use CRAN's archive repository for installing packages
     - > options(repos=c(CRAN="https://cran-archive.r-project.org/"))
   - Install PODIUM
     - Replace <path> in the command below with the path to the location where you downloaded the latest release of PODIUM
     - > install.packages("<path>/podium_1.2.0.tar.gz", type="source", repos=NULL)
   - Install PODIUMmethods
     - > install.packages("<path>/podiumMethods_1.2.0.tar.gz", type="source", repos=NULL)
   - Note that the version numbers in the file names will be different if downloading a newer release

Everything should now be installed and ready to run. The next section will explain how to format and structure data for use with PODIUM.

Linux and Mac Users

Linux and Mac users can skip step 3 in the installation instructions. However, it is important that any tools required for building R packages for your particular platform be installed. To install R 3.0.2, please follow the instructions on CRAN for installing an older version of R for your platform. Alternatively, you can download an build R 3.0.2 from source using the following link: https://cran.r-project.org/src/base/R-3/R-3.0.2.tar.gz

Unpack the archive and follow the instructions in the included INSTALL file to build and install R 3.0.2. This has been tested successfully on CentOS 7.7 (core).

General Overview of the PODIUM Workflow

The first step in the analysis routine involves a call to xcmsSet2( ) which reads in raw data and uses the findPeaks.CentWave algorithm to generate a list of peaks for each sample in the raw data

(Tautenhahn et al., 2008). The retention times are then aligned across all samples using the function retcor.obiwarp( ) which utilizes a principle known as dynamic time warping to correct for retention time variations across samples in a highly accurate and sensitive manner (Prince and Marcotte, 2006). Once aligned, the function group( ) is called which groups common peaks observed across multiple samples. Any peaks that do not occur frequently enough to be considered a reproducible biological feature are not grouped, discarding them from the rest of the analysis. This reproducibility criterion can be determined as a peak either needing to occur in a specific number or a specific fraction of the samples within at least one sample class. After grouping, the removeGroups( ) function is optionally called to remove any groups occurring before a user-defined retention time parameter. This allows for the initial column elution period to be removed from the analysis as it can often lead to spurious results. The fillPeaks( ) method is then called which fills in any 'missing' data for samples that do not have detected peaks for a given peak group. It does this by taking the information from existing peaks found in other samples belonging to a particular group and then integrates the profile matrix (an approximation of the raw data) in the same region for samples that have no detected peaks. This step is necessary in order to ensure that each peak group has a full set of intensity data from all samples, a critical requirement for any subsequent statistical analysis. The filled peak groups are then sent through the peak pairing algorithm with a call to labeledPeaks_groupPairing_tTest( ) which looks for peak pairs among the grouped peaks that represent labeled compounds. Finally, the identified peak pairs are passed to group_validationOutput( ) which prints extracted ion chromatograms (EICs) and mass spectra (MS) for each peak pair, as well as a final data table. This allows the user to quickly and easily validate any peak pair they desire without having to manually inspect the raw data. If the user has elected to run the program in "comparison mode", which compares two sets of samples to one another, a final statistical report is generated comparing the levels of each peak pair between the two sets. The report includes fold changes and p-values for each labeled peak pair.

Peak Pairing Algorithm

The peak pairing algorithm has been designed to be flexible and takes several user-defined parameters including the number of labeled atoms expected (nLabel), the expected mass difference between labeled and unlabeled atoms in Da (mLabel), a mass tolerance parameter in parts-per-million (mzppm), a retention time tolerance parameter in seconds (rterror), and an alpha value parameter (α) for determining thresholds of significance. Using these parameters, the algorithm first calculates the expected mass difference (Δ) between labeled and unlabeled isotopologues using the following formula: $\Delta = nLabel * mLabel$. The algorithm then takes each peak group and looks for any other peak groups that fall within ± rterror of the group's own retention time. It then searches within those groups for any whose mass is within ±mzppm of the expected isotopologue mass as defined by the following equation.

$$mzwindow = M + \Delta \pm (M + \Delta) * mzppm * 10^{-6}$$

Where *M* is the m/z value of the peak group used to define the search. Any valid matches (those that meet both the aforementioned mass and retention time requirements) are subjected to a one tailed t-test with equal variance on a per-sample-class basis. This t-test determines if there is a significantly greater signal for each putative labeled peak group in the label-fed samples vs the

non-label-fed samples within each sample class. A peak pair is returned as valid as long as the labeled peak group has $p \leq \alpha$ in at least one sample class (which, in the paper, was determined by genotype). The valid peak pairs are then concatenated into clusters. A cluster represents a related set of labeled peak pairs (eg. M, M+Δ, M+2Δ) due to the incorporation of multiple labeled moieties into one molecule. This is accomplished by searching the set of all peak pairs for "primary" peaks: peaks that are only found as unlabeled/base peaks across all peak pairs, never occurring as the labeled/+Δ peak. Clusters are formed by taking a primary peak and searching the set of all peak pairs to see if its +Δ peak ever forms the base peak of another peak pair. If it does, the +Δ peak of the new peak pair is added to the cluster and the process is repeated, searching again to see if the new +Δ peak forms the base peak of another peak pair. This process of serial concatenation continues until no peaks are found that can extend the cluster.

By applying this approach to all of the detected peak groups, the algorithm is able to identify all of the labeled peak pairs in the LC-MS data and, by consequence, all of the labeled (and in the case of this experiment, Phe derived) metabolites present in the samples being analyzed. After this initial identification process is complete, a parallel path is computed to aid in parameter optimization. The parallel path consists of searching for a +2Δ mass for each peak instead of the usual +Δ mass. The parallel path is then compared to the peaks from the original (serial) path that also exhibit a +2Δ relationship. If the results from the two paths are not equal, this is an indication that either the mzppm or rterror may need to be adjusted. Both paths are also checked for duplicate peaks (peaks that occur multiple times as either the base or +Δ peak). If there are more than a few duplicated peaks, this is an indication that either the mzppm or rterror may need to be reduced or that the peak picking and/or grouping parameters may need to be relaxed to avoid splitting peaks into multiple peak groups. While it is often not possible to have both the +Δ and +2Δ paths agree completely, a set of parameters that maximizes the agreement between paths while minimizing the number of duplicate peaks is considered to be optimal.

For more information on setting and running the peak pairing algorithm, please see the "Full details of custom/modified functions" section below.

Additions and Modifications from XCMS

In addition to the functions that handle the analysis of labeled compounds, PODIUM includes other useful additions and modifications to the base XCMS package. One of the main features of PODIUM is the introduction of the xcmsSet2 class. This class is an extension of the xcmsSet class and has been designed to hold all of the raw data in memory, eliminating the need to repeatedly read data from disk during runtime. This greatly increases the speed of some algorithms at the cost of using more RAM. Most algorithms from the base XCMS package have been updated to take advantage of the new xcmsSet2 class. The class also has a conversion function, convert( ), that will convert an xcmsSet2 object into a vanilla xcmsSet object. This method is especially useful for compatibility with downstream analysis packages, such as CAMERA, that require an xcmsSet object as input.

Another new addition is the function diffreport2( ). This function serves as a wrapper to the XCMS diffreport function, removing any data not part of a labeled peak pair before calling diffreport( ).

This allows for the differential analysis of only groups that were part of a peak pair, rather than all detected groups, as would normally occur for diffreport( ).

PODIUM also alters the way EICs are generated, allowing them to be created using a ppm-based m/z expansion method and altering the way retention time windows are calculated, symmetrically expanding them from the boundaries of each peak group rather than their centers.

Finally, PODIUM includes an enhanced version of the fillPeaks( ) method. This method contains modifications that address an issue in XCMS that would cause some filled peaks to have an intensity value of 0 or 'NA' instead of an appropriate value derived from the raw data. For more details on the issue and its fix, please see[4][5][6].

Further descriptions of these methods and their parameters can be found in the "Full details of custom/modified functions" section below.

Initial Setup

Before running PODIUM, it is important to note that, due to the nature of the statistical tests involved, we strongly recommend a minimum of three replicates per experimental group. In order to analyze data using PODIUM it must be converted to one of the following open formats: NetCDF (incl. AIA/ANDI) (UCAR/Unidata, 2019), mzXML, or mzData. Software packages from many vendors include the ability to export data into at least one of these formats. Alternatively, open-source conversion software, such as ProteoWizzard's msconvert tool, is freely available and can be used to convert many proprietary vendor formats. During the conversion, do not de-isotope or deconvolve the data as this can remove some of the peaks that arise from labeled molecules. Centroiding the data, however, is required for use with the peak picking algorithm findPeaks.centWave( ).

Once converted, the data then need to be organized into a hierarchical folder structure based on experimental groups. All folders must be contained within the same top-level folder, referred to as the "run" or "master" folder. Within this master folder, the organization of samples into subfolders will determine the sample classes that they are assigned to by PODIUM. For example, if comparing two sets of mutant plants to one another, you may want to put them into two different directories; "MutantA" and "MutantB". Within each directory you would then need to put two folders named "12C" and "13C". Within these two folders, you would put the label-fed samples into "13C" and the non-label-fed samples into "12C". This will create four sample sub-classes: MutantA.12C, MutantA.13C, MutantB.12C, and MutantB.13C representing MutantA unlabeled/labeled and MutantB unlabeled/labeled respectively. These sub-classes will be treated as two sample classes by PODIUM: MutantA and MutantB. It is important that all samples be placed into folders labeled "12C" and "13C" for each group, as PODIUM relies on these names to determine which samples are labeled and which are not.

[4] Initial issue: https://github.com/sneumann/xcms/issues/2
[5] Further discussion and solution: https://github.com/sneumann/xcms/pull/3
[6] Solution in the latest version of XCMS: https://github.com/sneumann/xcms/issues/130

The folder structure can contain as many layers as desired as long as it is level (meaning that all samples reside at the same depth within the folder structure). Building off of the previous example, a more complex experimental design might include time points for each mutant. In this case, each "Mutant" folder would contain subfolders named for each time point, and each of those subfolders would contain "13C" and "12C" folders filled with their respective labeled and unlabeled samples. This would result in sample sub-classes such as MutantA.T1.12C, MutantA.T1.13C, MutantA.T2.12C, MutantA.T2.13C, …, MutantB.T1.12C, MutantB.T1.13C, … which would cause PODIUM to treat each class of samples as its own group (e.g. MutantA.T1, MutantA.T2, MutantB.T1, MutantB.T2, ect.). Sample classes are referred to throughout PODIUM as "phenotypes" and their sub-classes can be viewed by calling sampclass( ) on the xcmsSet2 object or accessed via the @phenoData slot.

Once the data have been partitioned into the appropriate folders, you are ready to perform an analysis. We recommend using the provided driver script as it will automatically parse and analyze your data, providing diagnostic plots and tables at every step of the process. In addition, a final xcmsSet2 data object is also created that can be used for further analysis or converted to an xcmsSet object for compatibility with downstream analysis packages such as CAMERA. If you desire to perform your own custom analysis, we recommend the following workflow which closely follows the provided driver script.

- Peak picking using the xcmsSet2( ) constructor with method="centwave"
- Sample alignment and retention time correction via retcor.obiwarp( )
- Peak group detection via group( )
- Removal of groups that fall within the initial column elution period via removeGroups( )
- Filling in missing peaks via fillPeaks( )
- Labeled peak pair detection via labeledPeaks_groupPairing_tTest( )
- Printing of diagnostic plots and the final data table via groupValidationOutput( )
- Running diffreport2( ) if doing a two-sample comparison

Driver Script Setup

A driver script has been provided on the PODIUM GitHub repository that can completely automate a full PODIUM analysis. After setting the desired parameters, the user simply needs to source the script into a running R session by calling > source(<path to script/script.R>). If running in a unix-based environment, the script can also be run by putting the following on the first line of the script, !#<path/to/Rscript/install/Rscript> --vanilla , making the script file executable, and then executing it from a terminal.

Driver Script Configuration and Explanation of Parameters

This section describes how to configure the driver script and provides a detailed explanation of all parameters involved with the functions called by the driver script. Parameters will be described in the order that they appear in the script.

--Script-Specific Options and Parameters---

dataPath = <path>

dataPath should be set to the location of the top-level folder that contains all of the sub-folders with the data to be analyzed. See the "Initial Setup" section for more information on how to construct this folder and its contents.

resultsPath = <path>

resultsPath should be set to the location where all output should be stored. If the location does not exist, the script will attempt to create it.

Log = T or F

Enable logging? If set to T, a log file will be created that contains all console output (except for warning and error messages) generated over the course of a run. This will effectively document all important parameters/settings selected for each method and serves as a form of meta-data for each run.

Append = T or F

Append or overwrite log file if it already exists? If set to F, any pre-existing log file in the output directory will be overwritten.

datMin = 0, 1, or 2

Controls what data remains in memory after the analysis is complete. This is intended for use when the script is sourced in an interactive R session and the user desires to manually explore the results or perform a follow-up analysis after the script completes.

0: retain all data.

1: retain the initial, unprocessed data set as well as the final, completely processed data set.

2: only retain the final, fully processed data set.

saveData = 0, 1, or 2

Controls what data is saved to disk at the end of the run. This is primarily useful if you want to try several different sets of parameters for a single data set. By saving the dataSet object to disk, the initial peak picking step does not need to be repeated when experimenting with different parameters. This usually is a huge time savings as the peak picking step is by far the most time consuming. Depending on the setting, the following will be saved to disk:

dataSet: an xcmsSet2 object that only contains the results of the peak picking step. This object can serve as the starting point for multiple analyses that explore different sets of

parameters and is what is loaded if readRawData=F. Saved to the output directory as dataSet.RData

dataSet2: an xcmsSet2 object the contains the results from running the entire script. This object is best used to further explore the results of a run. Saved to the output directory as dataSet2.RData

env: the contents of the R environment after running the script. Contains all relevant parameters and setting to reproduce the analysis (but does not include dataSet or dataSet2). Saved as env.RData

0: save nothing to disk

1: save dataSet and env

2: save dataSet2 and env

3: save dataSet, dataSet2, and env

readRawData = T or F

Controls whether raw data is read and peak picking is performed. Requires dataSet.RData to already be present in the output directory if set to F (usually generated from a previous run). This functionality allows for testing different sets of parameters on the same data set without needing to repeat the peak picking procedure.

T: read raw data from disk and perform peak picking

F: attempt to read dataSet.RData from the output directory and skip the peak picking step. An error will occur if dataSet.RData does not exist.

comparison = T or F

multi = T or F

Comparison and multi control the script's run mode.

single/batch mode: If both comparison and multi are F, the script will run in single/batch mode. In this mode if a single phenotype (i.e. sample class) is present, it will be analyzed. If multiple phenotypes are present, each one will be analyzed individually in sequential order. Each phenotype will have its results generated in its own subdirectory within the specified output directory.

comparison mode: If comparison is T and multi is F, the script will run in comparison mode. In this mode, two phenotypes are analyzed simultaneously and diffreport2 is called to perform a differential expression analysis between the two phenotypes based on all detected labeled compounds.

multiple mode: If multi is T and comparison is F, the script will run in multiple mode. In this mode, all detected phenotypes are analyzed simultaneously. This mode is recommended for any sort of analysis that seeks to make comparisons across multiple sets of phenotypes/sample classes as comparisons are most accurate when all samples have been aligned, grouped, filled, and analyzed together.

selectedPhenos = c(<phenos>) or c("All")

Controls which sample classes/phenotypes to test. Must be a list of strings matching the detected sample classes. Sample classes can be view via sampclass(xcmsSet2). Note that the .12C and .13C parts of sample class names should not be included. If set to a list containing the single string "All", all detected phenotypes will be analyzed. This parameter is automatically ignored if running in single/batch mode.

val = "into", "intb", or "maxo"

Value to use for quantifying peak intensities.

into: integrated peak area

intb: baseline-corrected integrated peak area

maxo: maximum peak intensity

diffval = "into", "intb", or "maxo"

Value to use for quantifying peak intensities when generating a diffreport. Defaults to being the same value as val.

into: integrated peak area

intb: baseline-corrected integrated peak area

maxo: maximum peak intensity

Freport = T or F

Controls whether diffreport2 is called to generate a diffreport when running in comparison mode.

rem = T or F

rtmin = <numeric>

Controls whether to remove peak groups from the initial column elution period. If rem =T then any peak groups with a median retention time ≤ rtmin will be removed from the analysis.

nslaves = <integer>

The number of slave processes to be used for all parallelizable methods. It is recommended to use N-1 slaves, where N is the number of process threads available.

--Analysis Parameters--

This section contains the parameters and settings used in the paper along with descriptions of their effects.

Setup parameters

readRawData = T

Read in raw data files rather than data from a previous run

comparison = F

Do not run in comparison mode (for two phenotype comparisons only)

multi = T

Run in multi mode. All phenotypes will be analyzed simultaneously

selectedPhenos = c("All")

Analyze all detected phenotypes (rather than a pre-specified subset)

val = "into"

Use integrated peak area for quantification

Freport = F

Do not run diffreport (automatically disabled when analyzing more than two phenotypes)

rem = T

Remove peak groups from the initial column elution period

rtmin = 110

Define the end of the initial column elution period as 110 seconds. In conjunction with rem=T, this will cause any peak groups with median retention time ≤ 110 sec to be removed.

Peak picking parameters (passed to findPeaks.CentWave)

ppm = 40

The size of the m/z slice to use for region of interest (ROI) detection. The smaller the ppm, the smaller each mass slice is when looking for potential peaks

peakwidth = c(4,20)

The minimum and maximum expected peak width in seconds. Note: if the minimum width is too small it can lead to large peaks being split due to fluctuations or shoulders in the peak shape.

snthresh = 5

Signal to noise threshold for peak detection

prefilter = c(3,150)

Used to reduce the ROI checked for peaks. For example, c(3,150) indicates that a putative peak must have a maximum observed intensity over 150 count in at least 3 scans to be considered a region of interest and saved for the peak detection step.

integrate = 1

Used to integrate peak area and determine chromatographic peak boundaries. If set to 1, the descent to determine peak area and boundaries is done on the mexican hat function fit to the raw peak data (ie. integration based on the fit curve). If set to 2, the descent is performed on the raw data.

nslaves = <integer>

The number of slave processes to be spawned for parallel peak picking. It is recommended to use N-1 slaves, where N is the number of process threads available.

Retention time alignment parameters (passed to retcor.obiwarp)

response = 100

Controls how forcefully samples are aligned. Setting response to 100 will be the most forceful alignment, while 1 will produce a linear alignment

Peak grouping parameters (passed to group.density)

Note: The group( ) method considers what has been up to this point called a sample sub-class as sample classes. This means that group( ) will consider ".12C" and ".13C" forms of the same sample class as distinct sample classes.

minfrac = 0

Minimum fraction of samples in a sample class in which a peak must be present to be considered reproducible.

minsamp = 3

The minimum number of samples in a sample class in which a peak must be present to be considered reproducible.

Note: minfrac and minsamp are applied at the same time with the more stringent parameter taking precedence. If only one criterion is desired, then set the other parameter to 0.

mzwid = 0.01

The width in m/z to use for the overlapping mass window. We recommend calculating the mzwid based on the ppm parameter used for findPeaks.centWave and the median of the mass range measured by the mass spectrometer used to generate the data using the following formula: $mzwid = M * ppm/(2 * 10^6)$ where M is the median of detector's mass range.

bw = 2

The bandwidth of the gaussian kernel used to group peaks of similar retention time in each m/z slice

max = 50

The maximum number of peak groups that can be found per m/z slice

Peak filling parameters (passed to fillPeaks.chrom)

expand.mz = 1

Expansion factor for the integration area in the m/z dimension

expand.rt = 1

Expansion factor for the integration area in the rt dimension

min.width.mz = 1

Requires the integration area in the m/z dimension to be at least min.width.mz wide, widens the integration area if it is not

min.width.rt = 1

Requires the integration area in the rt dimension to be at least min.width.rt wide, widens the integration area if it is not

Peak pairing parameters (passed to labeledPeaks_GroupPairing_tTest)

mzppm = 10

The ppm used to generate the m/z tolerance window used for mass matching. The window is symmetric, so for 40 ppm the window is M±40 ppm*M where M is the mass being searched for.

mzabs = 0

An absolute m/z value to symmetrically expand the ppm based window by (ie ppm-window ±mzabs)

rterror = 2

The symmetrical retention time window in seconds to use for identifying co-eluting peaks. (e.g. rterror=2 yields a ±2sec rt window centered on the rtmed of each peak group)

alpha = 0.05

The alpha value to use for t-test validation of paired peaks

nLabel = 6

The number of occurrences of the label element to search for (i.e. 6 indicates to look for a mass difference of 6*mLabel)

mLabel = 1.0033548

The mass difference between the labeled and unlabeled element. In the paper, this was the mass difference between ^{12}C and ^{13}C

Output parameters (passed to group_ValidationOutput)

eicppm = 15

The ppm to use for calculating m/z windows during EIC generation

type = “valid”

The type of peak pairs to output diagnostics for. If “valid” all peak pairs identified as labeled will be used. If “suspect” peak pairs that failed the t-test stage of the peak pairing algorithm will be used instead.

Full Details of Custom/Modified Functions

removeGroups

removeGroups(xcmsSet2, rtrng = c(rtmin, rtmax))

Takes an xcmsSet or xcmsSet2 object and removes all groups whose median retention time falls within rtmin and rtmax (inclusive). Returns the input xcmsSet or xcmsSet2 object with the groups removed.

xcmsSet2: an xcmsSet or xcmsSet2 object with peak group data

rtrng: A numeric vector containing two numbers indicating the minimum and maximum bounds of the retention time window in seconds.

labeledPeaks_groupPairing_tTest

labeledPeaks_groupPairing_tTest(xcmsSet2, nLabel = 6, mzppm = 15, mzabs = 0, rterror = 1,

resultspath = NULL, phenoTag = NULL, phenotypes = NULL, alpha = 0.05,

value = "maxo", mLabel = 1.0033548, overide = F)

Finds all labeled peak-pairs as specified by the nLabel and mLabel parameters using the algorithm described above in the “Peak-Pairing Algorithm” section. The full equations for calculating the m/z search window is:

$$\Delta = nLabel * mLabel$$

$$mzwindow_{upper} = M + \Delta + (M + \Delta) * mzppm * 10^{-6} + mzabs$$

$$mzwindow_{lower} = M + \Delta - (M + \Delta) * mzppm * 10^{-6} - mzabs$$

Where M is the median m/z for the peak group used to generate the search query (i.e. the base/unlabeled peak). All comparisons and filtration are done using the median m/z and median rt of each peak group. Intensity values are generated using the XCMS function groupval( ) with method="medret". This results in intensity values being taken from each sample from the peak in that sample whose retention time is the closest to the median retention time of the peak group.

For a description of the files output by this method, please see the "Peak-Pairing Algorithm Output" subsection of the "Explanation of all output files" section below.

Key Parameters

xcmsSet2: an xcmsSet2 object with filled peak group data

nLabel: <integer> the number of occurrences of the label element to search for

mLabel: <float> the mass difference between the labeled and unlabeled element.

mzppm: <float> the ppm used to generate the m/z tolerance window used for mass matching.

mzabs: <float> an absolute m/z value to symmetrically expand the ppm based window by

rterror: <float> the symmetrical retention time window in seconds to use for identifying co-eluting peaks.

$$rtwindow = rtmed \pm rterror$$

a**lpha:** <float> the alpha value to use for t-test validation of paired peaks

mLabel: <float> the mass difference between the labeled and unlabeled element.

phenotypes: a character vector containing the names of all phenotypes/sample classes to use while performing peak pairing

Housekeeping Parameters

resultspath: path to the output directory

phenoTag: a string used to describe the current phenotype(s) being analyzed

value: Value to use for quantifying peak intensities.

into: integrated peak area

intb: baseline-corrected integrated peak area

maxo: maximum peak intensity

overide: force the method to run even if the xcmsSet2 object has not been filled (**not** recommended)

group_validationOutput

group_ValidationOutput(xcmsSet2=NULL, pheno=NULL, resultsPath=NULL,

type=c("valid","suspect"), value=NULL, clusters=NULL,

Indx=NULL, ppm=NULL, rtrng=25, overide=F, tag="NONE", nLabel=6)

Generates diagnostic EIC and MS plots for each cluster of labeled peaks using data from all samples and saves them to disk. Also outputs the final peak table that contains a full set of information for all detected peak pairs.

EIC and MS plots are saved to a subdirectory that is created within the output directory. The directory is named based on the input parameters as follows:

<type>_tTestFilter_<pheno>_nLabel_<nLabel>_<value>_<eicMethod>_<ppm>

This allows for multiple sets of diagnostic plots to be generated using different parameter settings. If a directory already exists with the same name as that dictated by the current parameter settings, the method will delete the folder and regenerate all plots.

For both the EIC and MS plots, lines labeled with "12C" are plotted from unlabeled samples while lines labeled with "13C" are plotted from labeled samples.

For the EIC plots, lines labeled only "12C" or "13C" are EICs generated for the base/unlabeled peak in the cluster. Lines labeled "…+X" are EIC's generated for the +Δ peak in the cluster. Lines labeled "…+More" are EIC's generated for any peaks in the cluster that are a multiple of Δ (i.e. +2Δ, +3Δ, …). The solid portion of each line indicates the rt window over which the peak was detected, the grayed-out portion of the line is not part of the peak but is shown for context. The two gray vertical lines indicate the rt window over which the corresponding MS spectra for the cluster were generated.

For the MS, the vertical black dashed line indicates the detected m/z for the base/unlabeled peak. The vertical red dashed line indicates the detected m/z for the +Δ/labeled peak. The observed m/z difference between these two values is printed in the upper right-hand corner of each plot. The rt window used to gather data to generate the MS can be seen on the corresponding EIC plot for the same cluster and is represented as two vertical dashed gray lines.

Both the EIC and MS plot files use the following naming convention:

Cluster_<cluster#>_Groups_<groupname1> <groupname2> … <groupnamex>_EIC/MS.png

This allows the user to easily search the plot output directory for an EIC and MS of interest, as the cluster number as well as the names of all peak groups that are part of the cluster are included in the file name.

It is important to note that this method can only be run after the peak-pairing method has been run at least once. This is because this method reads in the results file written to the output directory by the peak-pairing algorithm. A "file not found" error will be thrown if the file does not exist in the output directory.

The m/z window for EIC generation is created by the following equations:

$$mzwindow_{upper} = M + M * ppm * 10^{-6}$$

$$mzwindow_{lower} = M - M * ppm * 10^{-6}$$

Where M is the mean m/z across all peaks in a group. The retention time window over which each peak boundary is drawn is calculated using the following equation:

$$rtmin = rtmin - \frac{rtrng}{2}$$

$$rtmax = rtmax + \frac{rtrng}{2}$$

Where rtmin is the minimum rt for each peak and rtmax is the maximum rt for each peak.

Important Parameters

xcmsSet2: an xcmsSet2 object with filled peak group data

ppm: the ppm used to calculate the m/z window for EIC generation. If NULL, the default XCMS method for calculating the m/z window ("mzMinMax") will be used instead

rtrng: the full width of the symmetric rt range expansion to perform on each peak

type: "valid" or "suspect" the type of peak pairs to generate output for

"valid": generate output for all labeled peak pairs

"suspect": generate output for all peak pairs that failed the t-test stage of the peak-pairing algorithm (useful for diagnostic purposes)

clusters: a numeric vector containing the numbers of all clusters that should be included in the output

Indx: the indices of all groups to be included in the output, this should correspond to the indicies of all groups contained in the clusters selected for output unless the exclusion of certain groups is desired

Housekeeping Parameters

resultspath: path to the output directory

pheno: a string used to describe the current phenotype(s) being analyzed

nLabel: the value of nLabel used for the peak-pairing algorithm

value: Value to use for quantifying peak intensities.

into: integrated peak area

intb: baseline-corrected integrated peak area

maxo: maximum peak intensity

tag: a string to be appended to the beginning of the name of all output tables. If equal to "NONE" nothing will be appended

overide: force the method to run even if the xcmsSet2 object has not been filled (**not** recommended)

diffreport2

diffreport2(object,resultsPath=NULL ,type, pheno,

class1=NULL, class2=NULL, filebase = character(),

eicmax = 0, eicwidth = 200, sortpval = TRUE, classeic =NULL,

value = c("into","maxo","intb"), dataVal = c("into","maxo","intb"),

metlin = FALSE, h = 480, w = 640, mzdec=2,ppm=50, ...)

Diffreport 2 serves as a wrapper for diffreport. It prefilters the input xcmsSet2 object so that it only contains information on the detected labeled peaks before being passed on to diffreport. Diffreport2 will also suppress the normal output table from diffreport and print a modified version that includes pertinent information for each labeled peak pair. Diffreport has also been modified to perform the same m/z and rt expansion as described for group_validationOutput( ).

It is important to note that this method can only be run after group_validationOutput. has been run at least once. This is because this method reads in the results file written to the output directory by group_validationOutput. A "file not found" error will be thrown if the file does not exist in the output directory.

New Parameters Compared to diffreport

resultspath: path to the output directory

type: "valid" or "suspect" the type of peak pairs to generate output for

"valid": generate output for all labeled peak pairs

"suspect": generate output for all peak pairs that failed the t-test stage of the peak-pairing algorithm (useful for diagnostic purposes)

pheno: a string used to describe the current phenotype(s) being analyzed

dataValue: Value used for quantifying peak intensities when calling group_validationOutput

into: integrated peak area

intb: baseline-corrected integrated peak area

maxo: maximum peak intensity

ppm: the ppm used to calculate the m/z window for EIC generation. If NULL, the default XCMS method for calculating the m/z window ("mzMinMax") will be used instead

Explanation of All Output Files

This section explains the nature of all output files as well as when they are generated during the PODIUM routine. Many of the peak and group files described below are generated for diagnostic purposes. If a compound you know to be present is not being detected, then these files can be used to interrogate the data at every major step in the analysis, hopefully allowing the user to determine at what step there compound/feature of interest is being missed.

Common Columns

Many files share the same sets of columns. These common columns will be described here and any columns unique to a given file will be discussed along with that file's description below.

Peak Files

These columns are common across all peak files. More information on these columns can be found by consulting the findPeaks.centWave documentation.

mz weighted (by intensity) mean of peak m/z across scans

mzmin m/z peak minimum

mzmax	m/z peak maximum
rt	retention time of peak midpoint
rtmin	leading edge of peak retention time
rtmax	trailing edge of peak retention time
into	integrated peak intensity
intb	baseline corrected integrated peak intensity
maxo	maximum peak intensity
sn	Signal/Noise ratio, defined as (maxo - baseline)/sd, where maxo is the maximum peak intensity, baseline the estimated baseline value and sd the standard deviation of local chromatographic noise.
egauss	RMSE of Gaussian fit
mu	Gaussian parameter mu
sigma	Gaussian parameter sigma
h	Gaussian parameter h
f	Region number of m/z ROI where the peak was localised
dppm	m/z deviation of mass trace across scans in ppm
scale	Scale on which the peak was localised
scpos	Peak position found by wavelet analysis
scmin	Left peak limit found by wavelet analysis (scan number)
scmax	Right peak limit found by wavelet analysis (scan number)

Group Files

These columns are common to all files that contain peak group information

mzmed	median m/z of peaks in the group
mzmin	minimum m/z of peaks in the group
mzmax	maximum m/z of peaks in the group
rtmed	median retention time of peaks in the group
rtmin	minimum retention time of peaks in the group
rtmax	maximum retention time of peaks in the group
npeaks	number of peaks assigned to the group

Peak File Descriptions

_AllPeaks.csv

This file is output after the initial peak-picking step and contains information on all detected peaks

_Aligned_AllPeaks.csv

This file is output after retention time alignment and contains information on all detected peaks

_Filled_Aligned_AllPeaks.csv

This file is output after retention time alignment, group removal, and peak filling have been performed and contains information on all remaining peaks.

Group File Descriptions

_InitialGroupings_AllGroups.csv

This file is output after grouping has been performed and contains information on all detected groups

_InitialGroupings_Under_<rtmin>sec_Groups_Removed.csv

This file is output after grouping and removal of groups that occurred in the user-defined retention time window has been performed and contains information on all remaining groups

Peak-Pairing Algorithm Output

nLabel<nLabel>_Clusters_GroupNamesOnly.csv

This file contains information on all of the clusters of paired peaks detected by the algorithm. It only contains the names of the groups that make up each cluster.

clusterNumber: the number assigned to a cluster of labeled paired peaks

12C: the group name of the unlabeled/base peak in the cluster

12C_mz: the m/z of the unlabeled/base peak

13C_X: The group name of a labeled peak

13C_X_mz: the m/z for the labeled peak

rtmedmin: the minimum median retention time across all peak groups in the cluster

rtmedmax: the maximum median retention time across all peak groups in the cluster

tTestPrefilter_13C_Mplus_greater_12C_Mplus.csv

This file contains the results of the t-test filtration stage.

pval_<phenotype>: these columns provide the p-value for the statistical tests performed within <phenotype>'s samples

<phenotype>_valid(a=<alpha>): these columns indicate whether $p \leq$ <alpha> for the samples within <phenotype>. 1 is true 0 is false

peakTypeX: These columns indicate the peak types being tested. The columns immediately to the left of these columns contain the raw data from each sample for the specified peak type. "13C_" indicates that only labeled samples are being considered while

"12C_" indicates only unlabeled samples are being considered. "_Mplus" indicates the peak being considered is the one with the +Δ mass. "_M" indicates the peak being considered is the base peak / the one with the unlabeled mass. E.g. "13C_Mplus" would indicate that only data from labeled samples at the +Δ peak mass was used.

nLabel<nLabel>_Clusters_MatrixForm.csv

This file contains information on all of the clusters of labeled paired peaks detected by the algorithm. It includes standard group metrics for each peak group in the cluster. This should be considered the primary output from the peak pairing algorithm.

Iso: A string that identifies what part of a cluster that peak group comprises

12C : the base/unlabeled peak of the cluster

13C_X: The +X peak of the cluster where X is some multiple of Δ

lowermz: the lower m/z bound of the search window for that group's +Δ peak

uppermz: the upper m/z bound of the search window for that group's +Δ peak

group_validationOutput Table

nLabel<nLabel>_<type>_Clusters_tTestFilter_<val>.csv

This file is the final output table for the PODIUM routine. It is identical to the primary output table from the peak-pairing algorithm except that it also includes raw data from all samples for each peak group in the file. This file should be used for any downstream analysis.

diffreport2 Table

<val><sampclass1>_vs_<sampclass2>_tTestFilter_FinalReport.csv

This file contains the results of the differential expression analysis between sampleclass1 and sampleclass2. For more information on how this file is generated, please see the differport documentation.

fold	mean fold change (always greater than 1, see tstat for which set of sample classes was higher)

tstat	Welch's two sample t-statistic, positive for analytes having greater intensity in class2, negative for analytes having greater intensity in class1
pvalue	p-value of t-statistic
anova	p-value of the anova statistic if there are multiple classes
metlin	A URL to metlin for that mass
...	one column for every sample class
Sample Names	integrated intensity value for every sample
...	one column for every sample

Comparison to Other Programs Capable of Detecting Isotopically Labeled Compounds in Untargeted MS Datasets.

The table below compares and contrasts PODIUM to previously published programs that are capable of detecting isotopically labeled MS-features in untargeted MS data. We include this table to aid prospective users in deciding which method best fits their experimental needs.

(Blank cells mean the parameter is unreported or unclear)	**PODIUM**	**DLEMMA (Feldberg et al. 2018, 2019)**	**mzMatchISO (Chokkathukalam et al., 2013)**	**X13CMS (Huang et al., 2014)**	**geoRge (Capellades et al., 2016)**	**MetExtract (Bueschl et al., 2012, Bueschl et al., 2017)**	**IsotopicLabeling (Ferrazza et al., 2017)**	**MISO (Dong et al., 2019)**
Experimental design and initial MS feature identification								
Ability to compare MS feature intensity between more than one lines/ genotypes	Yes- can compare > 2 genotypes	Yes- manuscript describes comparison between two genotypes		Yes- manuscript describes comparison between two genotypes	Yes- manuscript describes comparison between two genotypes			No
Requires matched light and heavy isotope fed samples	Yes	Yes	No	Yes	Yes	No	No	Yes
Accepts feeding a single heavy isotope	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Accepts feeding two or more distinct heavy isotopes	No	Yes	No	No	No	No	No	Yes
Requires XCMS (i.e. xset object) output as input	Yes- integrated into program	No	Yes	Yes	Yes	No	Yes	Yes- But xset input is not integrated into program
Identification of labeled peaks								
Statistical test used to identify peaks derived from heavy isotope	Unpaired t-test with equal/ pooled variance	Fold-change parameter only		Welch's t-test	Welch's t-test			Tukey's honest significance test
One or two tailed test	One				Two			Two
Peaks tested for statistical difference	Only paired peaks (i.e. coeluting peaks exhibiting requisite mass shift)			Only paired peaks	First tests every co-eluting peak for significant difference between labeled and unlabeled samples. Then tests whether the			First tests every co-eluting peak for significant difference between labeled and unlabeled samples. Then tests

					significantly upregulated coeluting peaks form a labeled/ unlabled peak-pair			whether the significantly upregulated coeluting peaks form a labeled/ unlabled peak-pair
Additional fold change parameter to filter for heavy isotope-derived MS features		Threshold of 4 for labeled/unlabeled intensity to call a labeled peak			A labeled feature has to be 1.2 fold higher in heavy isotope fed sample than "peak filled" feature (i.e. background) in sample fed with light isotope	No		A labeled feature that does not pass initial statistical threshold can also be 10-fold higher in abundance than equivalent feature in 12C-fed sample
Additional methodology to specifically locate multiply labeled MS features	Yes. All lower-multiple intermediates must also be present and labeled	No		Searches for any mass shift matching the incorporation of 2 or more heavy atoms. No upper limit on the number of potential incorporations. Detection of intermediates not required.	Searches for any mass shift matching the incorporation of 2 or more heavy atoms. Number of potential incorporations limited by mass of putative labeled peak. Detection of intermediates not required			No
Requires user to provide a list of MS features to target	No	No	No	No	No	No	Yes	No

REFERENCES

BUESCHL, C., KLUGER, B., BERTHILLER, F., LIRK, G., WINKLER, S., KRSKA, R. & SCHUHMACHER, R. 2012. MetExtract: a new software tool for the automated comprehensive extraction of metabolite-derived LC/MS signals in metabolomics research. *Bioinformatics,* 28**,** 736-8.

BUESCHL, C., KLUGER, B., NEUMANN, N. K. N., DOPPLER, M., MASCHIETTO, V., THALLINGER, G. G., MENG-REITERER, J., KRSKA, R. & SCHUHMACHER, R. 2017. MetExtract II: A software suite for stable isotope-assisted untargeted metabolomics. *Anal Chem,* 89**,** 9518-9526.

CAPELLADES, J., NAVARRO, M., SAMINO, S., GARCIA-RAMIREZ, M., HERNANDEZ, C., SIMO, R., VINAIXA, M. & YANES, O. 2016. geoRge: A computational tool to detect the presence of stable isotope labeling in LC/MS-based untargeted metabolomics. *Anal Chem,* 88**,** 621-8.

DONG, Y., FELDBERG, L. & AHARONI, A. 2019. Miso: An R Package for Multiple Isotope Labeling Assisted Metabolomics Data Analysis. *Bioinformatics*.

FERRAZZA, R., GRIFFIN, J. L., GUELLA, G. & FRANCESCHI, P. 2017. IsotopicLabelling: an R package for the analysis of MS isotopic patterns of labelled analytes. *Bioinformatics,* 33**,** 300-302.

HUANG, X., CHEN, Y. J., CHO, K., NIKOLSKIY, I., CRAWFORD, P. A. & PATTI, G. J. 2014. X13CMS: global tracking of isotopic labels in untargeted metabolomics. *Anal Chem,* 86**,** 1632-9.

PRINCE, J. T. & MARCOTTE, E. M. 2006. Chromatographic alignment of ESI-LC-MS proteomics data sets by ordered bijective interpolated warping. *Analytical Chemistry,* 78**,** 6140-6152.

SMITH, C. A., E.J., W., O'MAILLE, G., ABAGYAN, R. & SIUZDAK, G. 2006. XCMS: processing mass spectrometry data for metaboite profiling using nonlinear peak alignment, matching, and identification *Anal Chem,* 78**,** 779-787.

TAUTENHAHN, R., BOTTCHER, C. & NEUMANN, S. 2008. Highly sensitive feature detection for high resolution LC/MS. *BMC Bioinformatics,* 9**,** 504.

UCAR/UNIDATA 2019. Network Common Data Form (netCDF) version 3.6.2 [software]. Boulder, CO.