Processing data using LUMA: LCMS-Based Untargeted Metabolomics Assistant

Jonathan Mosley and Marina Evich

Version 1.1

I. Introduction The LCMS-Based Untargeted Metabolomics Assistant (LUMA) package is an automated data processing tool, bridging the outputs of XCMS and CAMERA to a comprehensive data matrix in the R environment for discovery-based studies. The intended users of this R package are LC-MS-based untargeted metabolomics practitioners with minimal to advanced experience in the R environment who process raw LC-MS data files in preparation for downstream statistical analysis. The LUMA package performs feature-reduction by minimizing single features which do not pass quality control (QC) checks and can negatively impact downstream analyses. LUMA contains self-contained functions (herein referred to as modules) that allow for rapid, automated workflows to perform these QC steps with minimal user input. Furthermore, to expedite manual data curation for potentially conflicting isotope and ion adduct annotations, data visualization is consolidated to a single graphic per metabolite group. This graphic contains all EIC plots and psSpectra from CAMERA and new correlation matrices and dendrograms for all features attributed to a single metabolite. Final processed metabolite data, containing normalized intensities and user-defined metadata, can be exported to worksheets which are directly formatted to a number of analytical tools including MetaboAnalyst, while retaining traceability in the R environment.

The following tutorial is a guide to testing the basic functionality of LUMA using an LC-MS fish liver dataset using RStudio. It requires R 3.5.1 or newer. Provided with this tutorial are the following: 1) two R scripts, install-LUMA-depend.R and test_workflow.R; and 2) one binary package, lcmsfishdata_0.1.0.zip, an accompanying R package containing LC-MS metabolomics data from fish livers, the raw files of which are housed at O:\Public\jmosl01. These comprise all the necessary code and files to complete this SOP.

II. Installation guide

A. Install the R package LUMA and dependencies by sourcing the provided script "install-LUMA-depend.R" in the R environment. Note, the package dbplyr must be updated to at least version 1.3.0. Also, there is a known bug when the package Formula is also loaded;

```
## [1] "Error in Formula <<- Formula <- Annotated.Library[, \"Formula\"] :"
```

```
## [1] "cannot change value of locked binding for 'Formula'"
```

To fix, unload the Formula package, or simply restart the R session before rerunning the "test_workflow.R" script.

```
detach(package:Formula, unload = TRUE)
```

B. Install the R package lcmsfishdata from Install Packages, install from: Package Archive File (.zip) using the provided "lcmsfishdata_0.1.0.zip".

III. Directory organization There will be three directories necessary for testing the LUMA R package: the working directory, the input file directory and the data directory.

A. Create the working directory on your local machine and paste the "test_workflow.R" script into the working directory. Create an R project in this local directory by opening RStudio and create new project from your local working directory.

B. The location of the input directory will depend on your specific workstation and where lcmsfishdata was installed. This location can be printed to the R console by using the following code in R:

```
input.dir <- system.file("extdata",
    "Annotated_library.csv", package = "lcmsfishdata",
    mustWork = T)
x <- unlist(gregexpr("/", input.dir))
keep <- substr(input.dir, 1, x[length(x)] -
    1)
keep
```

C. The data directory containing the data files is located on the O drive. Do not delete or modify these files: O:\Public\jmosl01\LUMA-Data

IV. Testing modules LUMA is organized into modules, each of which has a specific function. To test each of these modules, open the provided R script file "test_workflow.R" in the working directory and run one line of script at a time. After each module, record the output printed in the console, including any warnings or errors if they occur.

A. Remove all objects from the environment by running the first line of the script:

```
rm(list = ls(all.names = TRUE))
```

B. Load the relevant libraries:

```
library(LUMA)
```

C. Retrieve the location of the input file directory, described in III.B, for example, C:/Program Files/R/R-3.6.1/library/lcmsfishdata/extdata.

D. Initialize the workflow by running the first module. This module will open a series of dialog boxes for the user to select input files to process the blanks and sample data in both positive and negative ionization mode. Note, that the order of the dialog boxes may change. If the Formula package is loaded in your R session, you will likely get an error here (see Section II.A for details to resolve the error.)

```
InitWorkflow()
```

1. In the first dialog box, select the input directory, described in III.B.
2. Select "OK" to use the LUMA recommended data directory.
3. Select "OK" to process blanks first or `cancel` to process samples first.
4. Enter "1" or "Positive" in the console to process positive ionization mode first. Alternatively, select "2" or "Negative" to process negative ionization mode.
5. In the dialog box, select the data directory, described in III.C.

6. In the dialog box, select the Sample Class file (Sample_Class.txt). This file contains sample sex, class and n information as well as a flag for endogenous (control class).
7. In the dialog box, select the Sample Data file (Sample_Data.csv). This file contains metadata for each sample file to be used.
8. In the dialog box, select the Annotated Library file (Annotated_library.csv). This file contains annotation information such as name, formula, molecular weight and retention times for metabolites.
9. In the dialog box, select the Search Parameters file (Search_Parameters.txt). This file contains additional processing information including the void retention time, % CV cutoffs and additional flags, such as the option to keep or remove singletons, i.e., metabolites with only a single feature.
10. Select "OK" to use an existing XCMS objects file.
11. Select "OK" to use an existing CAMERA objects file.
12. In the dialog box, select the adducts file (primary_adducts_pos.csv if selected positive in step IV.D.4. or primary_adducts_neg.csv if selected negative in IV.D.4.)
13. In the dialog box, select the XCMS objects file (for example, XCMS_objects_Blanks_Pos if selected processing blanks in positive mode first).
14. Enter "2" in the console to select the 2nd sample as the XCMS center parameter. Hit "enter" to confirm all previous selections.
15. In the dialog box, select the CAMERA objects file (for example, CAMERA_objects_Blanks_Pos if selected processing blanks in positive mode first).

E. The next module will remove peaks that elute prior to the void volume for the LC system, set by the input file selected in IV.D.9.

```
CullVoidVolume(from.table = "From CAMERA_with Minfrac",
    to.table = "Trimmed by RT", method = "mz")
```

F. The following module will annotate the culled peak list from annotations in the Annotated Library file selected in IV.D.8.

```
AnnotatePeaklist(from.table = "Trimmed by RT",
    to.table = "Annotated")
```

G. The following module, ParseCAMERA, will parse the annotations generated by CAMERA and remove redundant ion adduct annotations using predefined rules and will result in generating the first output files (one for each ionization mode and sample type processed) used for further evaluation to eliminate conflicting annotations. The files will be named with a convention based on the ionization mode and sample type. For instance, for blank samples processed in positive ionization mode, the resulting output file from this module will be called 'Blanks_Pos_Validate Metabolite Groups.xlsx'.

- if running positive ionization mode data, use the ParseCAMERA module with the following arguments:

```
ParseCAMERA(from.table = "Annotated",
    to.table = "output_parsed", CAMERA.obj = "anposGa")
```

- if running negative ionization mode data, use the following:

```
ParseCAMERA(from.table = "Annotated",
    to.table = "output_parsed", CAMERA.obj = "annegGa")
```

H. The following module, CombineFeatures, will sum the isotopic and adduct peaks and then combines all features into a single metabolite group.

```
CombineFeatures(from.table = "output_parsed",
    to.table = "Combined Isotopes and Adducts")
```

1. This is where visualization comes in handy for understanding how features are combined into a single metabolite group.
2. The previous module, ParseCAMERA, has graphing functionality built into it that is not tested in this SOP. However, so that you may evaluate this very important part of QC during data processing, we have provided you with two example pdfs that were generated with LUMA from the lcmsfishdata.

```
file.copy(from = paste0(keep, "/Peaklist_Neg_CorrPlots-GlutamicAcid_clear.pdf"),
    to = "./Peaklist_Neg_CorrPlots-GlutamicAcid_clear.pdf",
    overwrite = TRUE)
file.copy(from = paste0(keep, "/Peaklist_Pos_CorrPlots-Serine_muddy.pdf"),
    to = "./Peaklist_Pos_CorrPlots-Serine_muddy.pdf",
    overwrite = TRUE)
```

3. After running the above code, use a file explorer to navigate to your working directory. Open each of the pdfs in that directory.
4. Both pdfs contain the same graphic layout: in the top left, you will see a correlation matrix for all features in that metabolite group (the name of the metabolite is in the filename). Features that have large, dark blue circles have statistically significant strong positive correlation across all study samples. Missing circles are not statistically significant. The top right panel contains a cluster dendrogram of the same features to provide complementary information to the correlation matrix. The bottom panels contain extracted chromatograms (bottom right) and an example mass spectrum (bottom left) for all features in the metabolite group.
5. The LUMA graphic (all four panels together) contains all the necessary information for a quick visual check that all of the features being combined into a single metabolite group actually belong to the same metabolite in the original study sample. Multiple features can arise from the same molecule in solution during the electrospray ionization (ESI) process used to generate this LC-MS data. For a comprehensive evaluation of how features in ESI form, see the article at https://onlinelibrary.wiley.com/doi/full/10.1002/mas.21551.
6. The LUMA graphic is designed to help the user optimize a set of parameters specific to each study. These parameters, called Corr.stat.pos and Corr.stat.neg, are contained in the Search_Parameters.txt file and are critical to set the appropriate threshold for combining features into a single group per metabolite. By carefully following a separate SOP designed to optimize there parameters (using the "Validate Metabolite Group.xlsx" files generated by LUMA), the user can ensure that high quality data is generated by LUMA.
7. In a "clear" case, all features clearly belong to the same metabolite. This is evident in the example pdf for Glutamic acid: all features are strongly correlated, cluster closely together, and are perfectly overlapping in the Extracted Ion Chromatograms. Moreover, all feature assignments, as seen in the MS image (bottom left) make good sense from an ESI analytical chemistry perspective (i.e., all isotopic and adduct features present are consistent with a structure containing carbon atoms and a hydroxyl group, and the presence of a deprotonated dimer is indicative of a high concentration metabolite).
8. In a "muddy" case, there will be features mixed in that clearly do not belong to the primary metabolite in the group. As is seen in the example pdf for Serine, there are features that do not correlate/cluster well with the serine features (334, 341 and 350), even though they may correlate/cluster with each other (i.e., features 963,939, and 984). This is a case where the

features seen in the MS (bottom left) at 106, 107 and 108 amu, nominal mass, belong to our primary metabolite Serine, whereas the other features clearly belong to some other compound and should not be grouped with Serine features. At the provided optimum value for Corr.stat.pos, these spurious features are removed while evaluating the CombineFeatures algorithms.

I. The next module, CullCV, will remove components with a coefficient of variation (CV) greater than the user-specified cutoff across the Pooled QC samples. Removing these variable components reduces overall instrument specific variation in the data. In this case, a CV cutoff of 30% is used. Note: running this module only makes sense when processing data samples (if selected "Cancel" in step IV.D.3.).

```
CullCV(from.table = "Combined Isotopes and Adducts",
    to.table = "Trimmed by CV")
```

J. The CullMF module then removes components with a minimum fraction of less than 80% within a specific class. Removing these components reduces overall biological variation per class in the data, as these are not considered reliable biomarkers for a given class. Note: running this module only makes sense when processing data samples (if selected "Cancel" in step IV.D.3.).

```
CullMF(from.table = "Trimmed by CV",
    to.table = "Trimmed by MinFrac")
```

K. Repeat steps D-J for the other samples until positive blanks, negative blanks, positive sample data and negative sample data have all been processed.

L. The CullBackground module then removes background components which have been identified from the positive and negative blank samples.

```
CullBackground(from.tables = c("Trimmed by MinFrac",
    "Combined Isotopes and Adducts"),
    to.tables = c("Peaklist_Pos_Solvent Peaks Removed",
        "Peaklist_Neg_Solvent Peaks Removed",
        "Peaklist_Pos_Solvent Peaks Only",
        "Peaklist_Neg_Solvent Peaks Only"),
    method = "monoMass")
```

M. The next module, SimplyPeaklists, will combine the two peak lists from the positive sample data and negative sample data into a single list. This module will generate a 'Peaklist_combined.csv' file which contains samples and intensities for each metabolite.

```
file.copy(from = paste0(keep, "/EIC_index_pos.txt"),
    to = "./EIC_index_pos.txt", overwrite = TRUE)
file.copy(from = paste0(keep, "/EIC_index_neg.txt"),
    to = "./EIC_index_neg.txt", overwrite = TRUE)
SimplyPeaklists(from.tables = c("Peaklist_Pos_Solvent Peaks Removed",
    "Peaklist_Neg_Solvent Peaks Removed"),
    to.table = "Peaklist_Combined",
    peak.db = "Peaklist_db")
```

N. The NormalizePeaklists module will next normalize the peaklist using the default normalization, set to unit normalize.

```
NormalizePeaklists(from.table = "Peaklist_Combined",
    to.table = "Peaklist_Normalized")
```

O. The FormatForMetaboAnalystR module will format the peaklist and generate the files necessary to be used with MetaboAnalyst for further analysis (see section VI). The module should generate three files: 'Peaklist_for_MetaboAnalyst.csv', 'Metabolite_Metadata.csv', and 'Sample_Metadata.csv'.

```
FormatForMetaboAnalystR(from.table = "Peaklist_Normalized",
    to.csv = "Peaklist_for_MetaboAnalyst",
    data.type = "pktable", anal.type = "stat")
```

P. The final module, FinalWorkflow, will finalize the LUMA workflow, closing the databases and generating a LUMA log for traceability (see section V).

```
FinalWorkflow(peak_db = peak_db, lib_db = lib_db)
```

Q. After running the final module, save the R Workspace image. Run sessionInfo() and copy and paste the output in the console to a Word document saved in the working directory as "session_info.docx". Be sure to follow the instructions in Section VII to submit your beta testing results to the LUMA developers.

```
sessionInfo()
```

V. Traceability in LUMA After running the FinalWorkflow module, the LUMA log will be printed to the console. This log contains the names of all tables inside the SQLite databases, which were saved during the LUMA workflow that was ran in this SOP. LUMA contains database querying functions that can pull these tables into the R environment for traceability. An example of each function is shown below. Check the R documentation for more details.

A. Reconnect to LUMA database:

```
temp_db <- connect_peakdb(file.base = gen_filebase(mzdatafiles = DataFiles,
    BLANK, ion.id, IonMode))
```

B. Read a table into the R environment:

```
mydf <- read_tbl(peak.tbls[1], peak.db = temp_db,
    asdf = T)
mydf
```

C. Write a table to the LUMA database:

```
write_tbl(mydf = mydf, peak.db = temp_db,
    myname = "Reproduced_tbl")
```

D. Check that the reading and writing process is reproducible:

```
mydf2 <- read_tbl("Reproduced_tbl",
    peak.db = temp_db, asdf = T)
identical(mydf, mydf2)

## Quick query of features in a table
LUMA:::get_features(mytbl = "Reproduced_tbl",
    peak.db = temp_db)
```

VI. Submitting results to MetaboAnalyst After completing the final module, you will have generated all of the necessary files to submit this dataset to MetaboAnalyst for further analysis. For the purposes of this SOP, we ask only that you verify the successful upload of the dataset into MetaboAnalyst (https://www.metaboanalyst.ca/). Any analysis performed by the beta tester in MetaboAnalyst is informative, but optional.

A. Go to the MetaboAnalyst website (https://www.metaboanalyst.ca/).

B. On the home page, select '»click here to start«'.

C. Select the "Statistical Analysis" module near the top of the module wheel.

D. In the "Upload Data" section, set the following parameters:

   1. For Data Type, choose "Concentrations".
   2. For Format, choose "Samples in rows (unpaired)".
   3. Use the "Choose File" buttom to upload the 'Peaklist_for_MetaboAnalyst.csv' file.
   4. Click "Submit"

E. The next window is the 'Data Integrity' check. After reading about the data processing information, click "Skip" (this data has already had missing value imputation performed).

F. On the 'Data Filtering' window, choose "None (less than 5000 features)" and click 'Proceed' (this data has already been filtered).

G. On the 'Normalization Overview' window, choose the following parameters:

   1. For Sample normalization, choose "None".
   2. For Data Transformation, I recommend you choose "None", though you may change this if you want.
   3. For Data scaling, I also recommend "None".
   4. Click "Normalize" (even if you selected all "None" options).
   5. You may choose to view the result. Once you are satisfied, click "Proceed".

H. If you can see the "Select an analysis path to explore" page, then the dataset has been successfully uploaded into MetaboAnalyst . Feel free to explore the various statistical analyses that MetaboAnalyst offers; however, any results you generate are beyond the scope of this SOP.

VII. Submitting beta-testing results to developers After completing the final module (and exploring the LUMA outputs), you will have generated several output files including Microsoft .csv and .xlsx files in your working directory. Compress the entire working directory into a zip file. Rename the .zip extension to .piz to prevent Outlook from blocking the file, and send it to evich.marina@epa.gov.

Thank you for your beta testing, and we hope you enjoyed our package!