LipidFinder 2 manual

Jorge Alvarez-Jarreta
lipidfinder@cardiff.ac.uk

1 Introduction

LipidFinder is an open-source Python workflow designed to facilitate further targeted lipidomics analysis. LipidFinder categorises and removes noise and artifacts from liquid chromatography/mass spectrometry (LC/MS) datasets, searches the outcome in different databases to obtain putative identification of lipids, and assigns them to a class based on the LIPID MAPS classification system. The software quickly distinguishes and quantifies lipid-like features from contaminants, adducts and noise in LC/MS datasets that have been pre-processed using XCMS [1, 2]. Although we advise users to use XCMS, LipidFinder accepts input files from other pre-processing software tools (e.g. SIEVE™ from ThermoFisher).

This Jupyter notebook will explain how to get your computer ready to use LipidFinder and will guide you through LipidFinder's workflow to process your data. Before continuing, note that your data needs to be high resolution MS (e.g. at least 60000ppm) and with long chromatography to separate isobaric lipids. The approach is not suitable for shotgun lipidomics, MS/MS or low resolution datasets. The demo data provided in the **tests** folder was generated with an Orbitrap Elite Mass Spectrometer, but the software is MS-platform independent. It is composed by 12 LC/MS runs of macrophage-like cells: first 6 samples from RAW cells (264.7) and last 6 samples from mouse wildtype (C57BL/6). Afterwards, the data was pre-processed with XCMS Online using the set of parameters Orbitrap II.

LipidFinder can be downloaded from GitHub or accessed online via LIPID MAPS: http://www.lipidmaps.org/resources/tools/lipidfinder. For questions or suggestions regarding LipidFinder please contact us at lipidfinder@cardiff.ac.uk.

1.1 Read before using LipidFinder for the first time

LipidFinder has been designed for extensive clean-up of LC/MS datasets where a high degree of artifact removal is desired (e.g. discovery lipidomics). ESI-high resolution MS experiments contain many spurious signals that can arise from diverse sources, including common contaminants, adducts, in-source fragments, etc. LipidFinder is devised to work primarily as an add-on to XCMS, focusing on the clean-up of MS data files which have already been pre-processed for peak alignment and integration. Removal of these artifacts results in significantly cleaner datasets that perform better in downstream statistical analysis pipelines.

1.1.1 Key points for users:

- LipidFinder shares some functionalities with XCMS (e.g. isotope removal or retention time correction), however these use different algorithms and perform differently. Thus, running these functionalities again in LipidFinder significantly improves the quality of XCMS datasets.
- LipidFinder includes extra functionalities specifically designed to improve artifact removal that are not in XCMS, including: contaminant, adduct and stack removal, solvent removal, mass reassignment and outlier correction.
- Extensive LC chromatography is essential with the LipidFinder approach to separate isobaric lipids which are a major complicating issue in lipidomics MS. This method is not suitable for "shotgun" applications.
- Qualitative and semi-quantitative comparison of lipids using high resolution MS is a powerful approach for screening all lipids, both unknown and known in an unbiased manner so long as it is used appropriately and its benefits and limitations are appreciated and acknowledged in full by the user. When correctly applied, it is an MS approach that is extremely powerful for lipid discovery and comparative profiling of biological samples.
- Nowadays, targeted MS/MS based methods can measure up to 500 or more known lipids, however, we know that lipidomes from mammalian cells and plasma can contain thousands, perhaps up to 5,000 or more per sample, with approximately 50% of these not appearing in any database (true unknowns). Thus there is huge potential for discovery of new lipids using untargeted approaches, something that cannot be accomplished using MS/MS. The LipidFinder approach is a hypothesis-generating screening tool designed specifically to clean up MS datasets that initially present with around 60K datapoints of which we estimate around 10% to be real lipids [3]. All observations of interest obtained using this method require rigorous validation using **(i)** manual examination of chromatographic data where significant differences are detected to ensure peak quality, followed by **(ii)** gold standard MS/MS methods, where the differences seen are for known lipids that can be purchased as standards.
- Database matches provided using LipidFinder are putative, and ions are assigned to putative lipid classes, not to actual molecular species because the isobaric nature of lipids makes this impossible.
- The LipidFinder approach is analogous to the older Affymetrix array methods in genomics, which also require strict validation using qPCR, etc. The approach is semi-quantitative, and reports on relative changes between datasets. Internal standards can be used if desired to calculate A/IS.
- Statistical analysis post-LipidFinder, which can be found on its online version, can be used to identify significantly-different lipids that then need to be followed up using targeted methods and fully validated, etc.

2 Configuring your computer

LipidFinder has been tested for Python 3.6.3. This doesn't mean it won't work in earlier versions, but you might get errors or significant differences in the results. Some computer's operating systems come bundled with Python, but it can also be downloaded and installed from the Python Software Foundation. The first step is to download LipidFinder's latest package file from GitHub link

2.1 Default installation

LipidFinder's package includes all the instructions to install all the dependencies required. The easiest way to install LipidFinder is to open a command prompt/terminal, go to the folder where the downloaded Wheel file is located, and run the following command:

```
pip install LipidFinder-2.*-py3-none-any.whl
```

2.2 Alternative option: Anaconda

Many users prefer to use Anaconda, an open-source distribution aimed to do Python data science and machine learning in Windows, Linux, and MacOS. To install LipidFinder, open an Anaconda prompt/terminal and run the following command:

```
pip install LipidFinder-2.*-py3-none-any.whl
```

Note: All the scripts include the `.py` extension that needs to be removed in Windows systems.

3 Pre-processing the input files

We have included a thorough manual on how to pre-process your input files to leave them ready for LipidFinder in the **docs** folder, in a PDF document named *Input_preparation_manual.pdf*.

4 Setting up your parameters

There are two different ways to set the parameters of each module of LipidFinder's workflow. The first one is to run the `config_params` script. This script requires as argument the module you want to configure:

```
config_params.py -m peakfilter
```

Additionally, if you already have a parameters JSON file, you can load its values instead of LipidFinder's defaults (see example below). Once launched, the process will guide you through a question-answering system to configure each parameter. At the end, the program will ask for the path and file name in which you want to save the new set of parameters:

```
config_params.py -m peakfilter -p my_parameters.json
```

The second option is through a Jupyter notebook (like this one). The *Configuration* module includes a graphical user interface (GUI) class to set up each parameter of the selected module interactively based on Jupyter's widgets. The following code shows an example of how to launch the GUI to set *Amalgamator*'s parameters based on default values:

```
from LipidFinder.Configuration.LFParametersGUI import LFParametersGUI
LFParametersGUI(module='amalgamator');
```

To use an existing parameters JSON file instead of the default values, you need to add the argument `src=x`, where `x` is the path to the JSON file, to the `LFParametersGUI()` call.

Hint: once you have configured *PeakFilter*'s parameters, you can use that JSON file as template for the other modules so you do not need to type in again the value of the parameters they all share

(e.g. *m/z* column name). *Warning:* parameter `firstSampleIndex` needs to be changed when using PeakFilter's summary output file as input.

We have included a **help** option to display the description, usage and other information of each Python script included in LipidFinder. For instance, for the previous script, the command to run would be the following:

```
config_params.py -h
```

4.1 Backwards compatibility

A user that has used LipidFinder 1.0 might be interested in repeating their experiments with the new version or run new ones under a similar parameter configuration. Thus, we have developed a script to transform the old parameters CSV file for *PeakFilter* and *Amalgamator* to the new parameters JSON files for the same modules. To run it you will also need the old adducts CSV file to update the lists of adduct pairs. We have included an example of these two files in the `tests` folder (available on GitHub) to illustrate how to use the script:

```
update_params.py -p tests/LipidFinder-1.0/old_parameters.csv \
    -a test/LipidFinder-1.0/old_adducts.csv -o results
```

The script will generate two files: `peakfilter.json` and `amalgamator.json`. Be aware that these new parameters JSON files are incomplete (some new parameters have been introduced in LipidFinder 2.0) and will raise an error when used for their corresponding module. They should be handled first by `config_params.py` (`-p` argument) to fill in the missing parameters and generate a complete version.

5 LipidFinder's workflow

LipidFinder's complete workflow is composed by three modules: PeakFilter, Amalgamator and MSSearch. We have developed one script for each one to ease their usage. Each module will create a `log` file (named after the module) that will save any information that might be useful for the user, e.g. which frames that have been removed by which stages during *PeakFilter*. A new run will append the new information at the end of the `log` file if it already exists, so no information is lost.

A standard LipidFinder workflow would first process the pre-aligned data with *PeakFilter* (once for negative and once for positive ion polarity), afterwards *Amalgamator* would merge both files' information based on matching *m/z* values, and finally, *MSSearch* would identify and classify lipid-like features with the selected LIPID MAPS database. Alternatively, LipidFinder can also process a single file with *PeakFilter* and run *MSSearch* afterwards.

The following examples are all based on the demo data pre-processed with XCMS, but we also provide an alternative to show LipidFinder's flexibility with SIEVE™ pre-processed files (just replace *XCMS* by *SIEVE* in each command).

5.1 PeakFilter

PeakFilter is intended to clean-up the data from contaminants, adducts, stacks and other artifacts like in-source ion fragments and salt clusters. Among its parameters, *PeakFilter* has several

"switches" for determined filtering functionalities that should be configured based on the experimental set-up that generated the input dataset.

In most scenarios, an experiment involving LC/MS will generate two sets of data with different ion polarity: one negative and one positive. After they have been pre-processed separately with XCMS, we need to process each file individually with PeakFilter. Using our demo data available on GitHub under the `tests` folder, we show first how to process the negative polarity CSV file:

```
run_peakfilter.py -i tests/XCMS/negative.csv -o results \
    -p tests/XCMS/params_peakfilter_negative.json
```

And then the positive one:

```
run_peakfilter.py -i tests/XCMS/positive.csv -o results \
    -p tests/XCMS/params_peakfilter_positive.json
```

By default, *PeakFilter* will generate the complete filtered file and a **summary** output CSV file with the relevant information of each remaining frame.

The output file names will always contain ion polarity, so running *PeakFilter* once for each polarity will not be a problem when choosing the same output folder (e.g. `results` in the previous examples). However, if we change the parameters and run *PeakFilter* again with the same output folder, we will overwrite any previous output file for the same polarity.

5.2 Amalgamator

Amalgamator merges the output files for both negative and positive ion polarities generated with *PeakFilter*. By default, it will keep every frame that exists in only one of the input files, and for those with a match in both files, *Amalgamator* will retain the information of the one with the highest mean intensity for all samples tagging the selected source in the output file's **Polarity** column.

```
run_amalgamator.py -neg results/peakfilter_negative_summary.csv \
    -pos results/peakfilter_positive_summary.csv \
    -p tests/XCMS/params_amalgamator.json -o results
```

Duplicates are identified by comparing the negative file with the positive file within a small retention time tolerance and a corrected *m/z* tolerance (negative *m/z* + 2H+, followed by negative *m/z* + H+ + CH3+ for phosphotidylcholine and sphingomyelins with phosphocholine head group). Any hits are classed as a match.

Alternatively, you can use the complete output files generated by *PeakFilter* as input files if you want to keep every column of your source data file.

5.3 MSSearch

MSSearch has been designed to identify and classify lipid-like features from either *PeakFilter* or *Amalgamator* output file, using the knowledge available in LIPID MAPS. The output file will include all the matches for each *m/z* value in the input file (within the indicated tolerance in the parameters JSON file). The output file will also include every frame not found in the selected database, and they will be classified as *unknown*. Finally, *MSSearch* will create a lipid-category scatter plot of the results by *m/z* and retention time in a PDF file (by default).

```
run_mssearch.py -i results/amalgamated.csv -o results \
    -p tests/XCMS/params_mssearch.json
```

References

[1] E. Fahy, J. Alvarez-Jarreta, C. J. Brasher, A. Nguyen, J. I. Hawksworth, P. Rodrigues, S. Meckelmann, S. M. Allen, and V. B. O'Donnell. LipidFinder on LIPID MAPS: peak filtering, MS searching and statistical analysis for lipidomics. *Bioinformatics*, 35(4):685–687, 08 2018.

[2] A. O'Connor, C. J. Brasher, D. A. Slatter, S. W. Meckelmann, J. I. Hawksworth, S. M. Allen, and V. B. O'Donnell. LipidFinder: A computational workflow for discovery of lipids identifies eicosanoid-phosphoinositides in platelets. *JCI Insight*, 2(7):e91634, 2017.

[3] D. Slatter, M. Aldrovandi, A. O'Connor, S. Allen, C. Brasher, R. Murphy, S. Mecklemann, S. Ravi, V. Darley-Usmar, and V. O'Donnell. Mapping the human platelet lipidome reveals cytosolic phospholipase a2 as a regulator of mitochondrial bioenergetics during activation. *Cell Metabolism*, 23(5):930 – 944, 2016.