Analysis of the association between fatigue and the plasma lipidomic profile of inflammatory bowel disease patients

Table of Contents

UPLC-MS analysis and peak table generation

Sample preparation. Plasma samples were allowed to thaw on ice. 150 µL cold CH3OH were added to 50 µL of plasma for protein precipitation. The mixture was homogenized (Vortex, 20 s) and centrifuged at 13000 x *g* and 15 °C for 15 min. 150 µL of supernatant were evaporated to dryness (SpeedVac) and dissolved in 60 µL of (1:1) (5:1:4 IPA:CH3OH:H2O, 5 mM CH3COONH4, 0.1% v/v HCOOH):(99:1 IPA:H2O, 5 mM CH3COONH4, 0.1% v/v HCOOH). A blank extract was prepared following the same procedure but replacing plasma with water. For quality control, 10 µL of each sample extract were pooled in a glass vial to create a QC sample.

Lipidomic analysis. Untargeted lipidomic analysis was carried out employing a 1290 Infinity HPLC system from Agilent Technologies (CA, USA) equipped with a UPLC BEH C18 column (50 x 2.1 mm, 1.7 µm) from

Waters (Wexford, Ireland). The flow rate was set to 400 µL min-1running a binary mobile phase gradient starting at 98% of mobile phase A (5:1:4 IPA:CH3OH:H2O 5 mM CH3COONH4, 0.1% v/v HCOOH) during 0.5 min followed by a linear gradient from 2 to 20% of mobile phase B (99:1 IPA:H2O 5 mM CH3COONH4, 0.1% v/v HCOOH) during 3.5 min and from 20 to 95% v/v of mobile phase B in 4 min; 95% v/v of mobile phase B was maintained during 1 min; return to initial conditions was achieved in 0.25 min and were maintained for a total run time of 14 min. Column and autosampler were kept at 55 and 4 ˚C, respectively, and the injection volume was 2 µL. For MS detection, an Agilent 6550 Spectrometer iFunnel quadrupole time-of-flight (QTOF) MS system working in the ESI+ and ESI- modes was used. Full scan MS data in the range between 70 and 1500 *m/z*were acquired at a scan frequency of 5 Hz using the following parameters: gas T, 200 °C; drying gas, 14 L/min; nebulizer, 37 psi; sheath gas T, 350 °C; sheath gas flow, 11 L min-1. Mass reference standards were introduced into the source for automatic MS spectra recalibration during analysis via a reference sprayer valve using the 149.02332 (background contaminant), 121.050873 (purine), and 922.009798 (HP-0921) *m/z* in ESI +, and 119.036 *(*purine) and 980.0163 (HP-0921+AcOH) in ESI-, as references. ESI+ and ESI- analysis were carried out in independent batches. Between each mode, the instrument was cleaned and calibrated according to manufacturer guidelines. Each sample batch included 55 plasma samples (23 fatigue, 24 non-fatigue and 8 additional samples excluded due to clinical criteria) in a randomized order, 12 QCs (1 QC every 6 samples, 2 at the beginning of the sequence and 2 after the injection of the last plasma sample), and 4 blanks (1 at the beginning and 3 at the end of the sequence). QCs were used to monitor the instrument performance, correct within-batch effects, and identify unreliable, background, and carry-over features as described elsewhere **[***Sci. Rep.* **2019**, *9* (1), 9822**][***Anal. Chim. Acta* **2018**. https://doi.org/10.1016/j.aca.2018.04.055**][***The Analyst* **2015**, *140* (22), 7810–7817**]**. A set of 9 QCs were injected at the beginning of each batch for system conditioning and MS/MS data acquisition. MS/MS spectra were acquired using the auto MS/MS method with the following inclusion *m*/*z* precursor ranges: 70–200, 200–350, 350–500, 500–650, 650–800, 800–950, 950–1100, and 1100–1200 from 70 to 1200 using, in all replicates, a rate of 5 spectra/s in the extended dynamic range mode (2 GHz), a collision energy set to 20 V, an automated selection of five precursor ions per cycle and an exclusion window of 0.15 min after two consecutive selections of the same precursor.

Peak table generation. Peak table generation was carried out using XCMS software15. The *centWave* method was used for peak detection with the following parameters: mass accuracy, 20 ppm; peak width, (3,15); snthresh, 12; prefilter, (5,3000). A minimum difference in *m/z* of 7.5 mDa was selected for overlapping peaks. Intensity weighted *m/z* values of each feature were calculated using the *wMean* function. Peak limits used for integration were found through descent on the Mexican hat filtered data. Grouping before and after RT correction was carried out using the *nearest* method and 9 s as *rtCheck* argument. Finally, missing data points were filled by reintegrating the raw data files in the regions of the missing peaks using the *fillPeaks*method. The CAMERA package was used for the identification of pseudospectra based on peak shape analysis, isotopic information and intensity correlation across samples. Each dataset was processed with the following CAMERA functions: *xsAnnotate*, *groupFWHM*, *findIsotopes*, *groupCorr* and *findAdducts*using standard arguments. Identification and elimination of uninformative features was carried for ESI+ and ESI- data sets independently.

Import and pre-process the XCMS ESI+ peak table

1. Download the zip files from Zenodo repository. Unzip the files in a local folder (e.g. C/../pr-2020-00462c_DataRepository)

2. This section describes how to import into the MATLAB workspace: the XCMS peak table (.csv file), the set of MSMS experimental data acquired during the analysis of QCs, a reference spectral library, and the sample meta-data (included in the MetaData .xls file). It also describes how to use the experimental MSMS and the spectral library for metabolite annotation, and how to attach the annotations generated using LipiDex. Finally, it described how the within-batch effect data corrrection and clean-up was performed.

```
clear
clc
path1 = '/Volumes/Macintosh HD/Users/2017 LEITAT/Research3/2020 IBD_Lipids/Manuscript_1
```

path1: local folder where the data are stored.

```
Folder.matlab = strcat(path1,'/matlab');
Folder.ms2_exp = strcat(path1,'/matlab/ms2/esip_ms2');
Folder.lcms_annotate = strcat(path1,'/LCMS_annotate');
Folder.qcsvrc = strcat(path1,'/LIBSVM_qcsvrc');
addpath(genpath(Folder.lcms_annotate))
addpath(Folder.qcsvrc)
cd(Folder.matlab)
```

Folder.matlab: working directory

Folder.ms2_exp: Local folder where the experimentally acquired ESI+ MSMS spectra are stored,

Folder.lcms_annotate: folder that includes the functions used for data import and annotation.

Folder.qcsvrc: folder that includes the functions used for within-batch effect correction using QC-SVRC.

Import ESI+ MSMS spectral library

```
load ms2_library_esip.mat % ms2 & info
```

The ms2_library_esip.mat file contains the MSMS spectra and the information of the metabolites. Please, open the ms2_library and ms2_library_info to get an overview of the information contained.

Import experimental ESI+ MSMS spectra

Import the experimentally acquired MSMS data into the MATLAB workspace. Define the folder where the MSMS chromatograms (**in .ms2 format**) are stored. The function will extract and import the full set of MSMS spectra into a single data structure (ms2_exp or MS2_exp).

- *This step migh take a few minutes, depending on the amonunt of data and the computer. You can skip the data-import by loading the MSMS data included in the ms2_exp_esip.mat file*.

```
ms2_exp = msms_exp_db(Folder.ms2_exp);
% or run the next line to directly import the msms data and save some time
% load ms2_exp_esip.mat
MS2_exp = ms2_exp_reshape(ms2_exp);
```

Import the xcms peak table (ESI+)

Import the XCMS peak table and the meta-data of each sample included in the 'MetaData.xlsx' file. The function will create a single data structure will the xcms data and related meta-data.

```
metadata_file = 'MetaData_IBD.xlsx'; % define the metadata file
xcms_file = 'xcms_IBD_ESIp.csv'; % define the xcms file (ESI+)
S0 = xcms2strc(xcms_file,metadata_file); % creates an structure with the same info as t
```

Data preprocessing (ESI+)

Metabolite annotation

Run metabolite annotation using the MSMS spectral library, the MSMS spectral data, and the XCMS information (m/z, m/z min-max values, RT, RT min-max values, pcgroup). There are different options that can be tuned depending on the quality of the experimental MSMS data and the spectral library used. In this case, we used the following parameters:

```
options_search.type_msms = {'all'};
options_search.collision_energy = {0,'>='};
options_search.list_adducts = {'ESI+'};
options_search.delta_rt = 0.15;
options_search.delta_mz = {20,'ppm'};
options_search.rt_weight = 1;
options_search.mz_weight = 1;
options_search.min_int = {[0.05 250],'%max-absolute'};
options_search.norm_msms = 'base peak';
options_search.isotop_13C = {[0.005,1],'n'};
options_search.min_ions_msms = [10 4];
options_search.min_ions_match = 4;
options_search.dotproduct = {0.7,'A&R','geomean'};
options_search.m = 1.2;
options_search.n = 0.9;
options_search.nr_ids = 1;
options_search.label_pcgroups = 'y';
options_search.match_msms = {'both'};
options_search.RTs = [];

results_id = lcms_annotate([S0.mz S0.rt S0.min_rt_xcms S0.max_rt_xcms S0.camera{(1:end)
    ms2_exp,ms2_library,ms2_library_info,options_search); % run lcms_annotate
```

- (opt.) some check of the annotated feats. Note: keep in mid that these functs. use the results_id.db_ID_class and the results_id.db_ID_name, results_id.db_ID, results_id.mean_dp, ...

```
plot_mzrt_annotated_class(S0,results_id,0) % plot a distribution of classes with >0 ann
plot_ms2_parents(MS2_exp) % plot the distribution of framgented feats in the m/z-RT spa
table_matches = find_annotated_metab_name(results_id,1,'DG'); % find features annotated
plot_db_match_strc(S0,5411,1,options_search,ms2_library,results_id,MS2exp);
% plot a spectral match: feature #5411 (best match)
```

Add annotations find using LipiDex to the results_id structure

LipiDex is a freely available software suite for lipid identification. It employs *in silico* fragmentation templates and lipid-optimized MS/MS spectral matching algorihms to identify lipid species and unknown compounds from diverse sample matrices. https://www.sciencedirect.com/science/article/pii/S240547121830108X

- LipiDex reference: Paul D.Hutchins, Jason D.Russell, Joshua J.Coon, LipiDex: An Integrated Software Package for High-Confidence Lipid Identification, Cell Systems, 2018, 6/5), 621-625.e5

This function will match the xcms features with the metabolites annotated using LipiDex included in the generated .csv results files (i.e. X20191120_003_Results.csv, X20191120_004_Results.csv,..., X20191120_009_Results.csv).

```
[results_id,table_lipids_id] = ...
    lipidex_2_strct(S0.mz,S0.rt,S0.camera{:,3},Folder.ms2_exp,0.15,...
    {20,'ppm'},[],[0 0 -1],results_id);
```

Create a field with IDs from both types of annotation

By running two different (but similar) annotation algorithms using different databases, we increase the potential number of annotated metabolites. Here, if a feature has been annotated using both, the 'HMDB/LipidBlast' and the 'Lipidex' libraries, the first one is kept.

A new field in the structure is created (results_id.dbs_name) with the information from both annotations.

```
[~, h] = find(~isnan(results_id.mean_dp(1,:))>0);
[~, l] = find(~isnan(results_id.lipidex_dp(1,:))>0);
hs = zeros(size(results_id.dp,2),1); hs(h) = 2;
ls = zeros(size(results_id.dp,2),1); ls(l) = 1;
ids = hs+ls; % ids: 3:id@lipidex & hmdb;  2:id@hmdb; 1:id@lipidex; 0:no id
results_id.dbs_name = results_id.db_ID_name(1,:); % keep the best match
for j = 1:size(results_id.dp,2)
    if ids(j)==1 % if ID only @lipidex
        results_id.dbs_name{1,j} = results_id.lipidex_ID_name{1,j};
    else
    end
end
clear h hs ids j l ls
```

Create a data set object (PLSToolbox)

This step is optional (you can keep using a data structure if the PLSToolbox is not available), but it is recommended. This function adds ID & classes etc info to a unique dso (**X0**)

```
X0 = annotation2dso(S0,results_id);
```

Within-batch effect correction and data clean-up

This step includes a within-batch effect correction using QCs and the QC-SVRC algorithm, followed by the identification and elimination of unreliable features (RSD(qc)>20%, or detected in blanks).

```
%% qc-svrc
addpath(Folder.qcsvrc)
qc = find(strcmp(X0.classid{1,1},'QC')>0);
min_nr_qcs = 9;
lod = -10;
epsilon_range = [2:0.5:5];
gamma_range = 10.^[0:5];
C_interval = 90;
kfoldcv = numel(qc);
klfolcv_iterations = 1;
[Rtemp] = qcsvrc_batch_2pc(X0.data,qc,min_nr_qcs,lod,...
    epsilon_range,gamma_range,C_interval,kfoldcv,klfolcv_iterations,'subtract','n');
X1 = X0;
X1.data = Rtemp.Xc;
skipped_svr = find(isnan(Rtemp.nSVs_opt));
if numel(skipped_svr)>0
    X1(:,skipped_svr) = []; % exclude features for wich qc-svrc could not be performed
else
end
results_id1 = results_id;
fn = fieldnames(results_id1);
for i = 1:numel(fn)
    results_id1.(fn{i})(:,skipped_svr) = [];
end
```

A new data set object (**X1**) is created with the corrected data, and the *results_id* is also modified accordingly to create the *results_id1*.

Then, exclude from the analysis those features with RSD(QCs)>=20 after QC-SVRC. A new data set object (**X2**) is created with the corrected data, and the *results_id1* is also modified accordingly to create the *results_id2*.

```
clear  Rtemp skipped_svr min_nr_qcs lod epsilon_range gamma_range ...
    C_interval kfoldcv klfolcv_iterations i fn

% rsd @qcs
k = 100*std(X1.data(qc,:))./mean(X1.data(qc,:));
k = find(k<20);
X2 = X1(:,k);
results_id2 = results_id1;
fn = fieldnames(results_id2);
for i = 1:numel(fn)
    results_id2.(fn{i}) = results_id2.(fn{i})(:,k);
end
clear k
```

Finally, the 'background/carry-over' features are excluded from the analysis: those for which the median values in QCs was lower than 6 times the median value in blanks. A new data set object (**X3**) is created with the corrected data, and the *results_id1* is also modified accordingly to create the *results_id3*.

```
% Blank clean-up
blank = find(strcmp(X2.classid{1,1},'BLANK')>0);
r = 6;
k = find(median(X2.data(qc,:)) >= median(X2.data(blank,:))*r);
X3 = X2(:,k);
results_id3 = results_id2;
fn = fieldnames(results_id3);
for i = 1:numel(fn)
    results_id3.(fn{i}) = results_id3.(fn{i})(:,k);
end
clear k r fn i blank
```

Define a new dataset object including only the annotated features:

```
annotated_features = find(~cellfun(@isempty,results_id3.dbs_name(1,:)));
Xa = X3(:,annotated_features);
fn = fieldnames(results_id3);
results_xa = results_id3;
for i = 1:numel(fn)
    results_xa.(fn{i}) = results_id3.(fn{i})(:,annotated_features);
end
```

And save the results

```
save ibd_esip.mat S0 X1 X2 X3 Xa results_id results_id1 results_id2 results_id3 results
```

Import and pre-process the XCMS ESI- peak table

This section describes how to import into the MATLAB workspace: the XCMS peak table (.csv file), the set of MSMS experimental data acquired during the analysis of QCs using ESI-, a reference spectral library, and the sample meta-data (included in the MetaData .xls file). It also describes how to use the experimental MSMS and the spectral library for metabolite annotation, and how to attach the annotations generated using LipiDex. Finally, it described how the within-batch effect data corrrection and clean-up was performed.

```
clear % clear the workspace
clc
path1 = '/Volumes/Macintosh HD/Users/2017 LEITAT/Research3/2020 IBD_Lipids/Manuscript_I
Folder.matlab = strcat(path1,'/matlab');
Folder.ms2_exp = strcat(path1,'/matlab/ms2/esin_ms2');
```

Folder.matlab: working directory

Folder.ms2_exp: Local folder where the experimentally acquired ESI- MSMS spectra are stored,

```
cd(Folder.matlab)
```

Import ESI- MSMS spectral library

```
load ms2_library_esin.mat % ms2 & info
```

The ms2_library_esip.mat file contains the spectra and the information of the metabolites.

Import experimental ESI- MSMS spectra

Import the experimentally acquired MSMS data into the MATLAB workspace. Define the folder where the MSMS chromatograms (**in .ms2 format**) are stored. The function will extract and import the full set of ESI-MSMS spectra into a single data structure (ms2_exp or MS2_exp).

- *This step migh take a few minutes, depending on the amonunt of data and the computer. You can skip the data-import by loading the MSMS data included in the ms2_exp_esip.mat file*.

```
%ms2_exp = msms_exp_db(Folder.ms2_exp);
 load ms2_exp_esin.mat % run this line to directly import the msms data
MS2_exp = ms2_exp_reshape(ms2_exp);
```

Import the xcms peak table (ESI-)

Import the XCMS peak table and the meta-data of each sample included in the 'MetaData.xlsx' file. The function will create a single data structure will the xcms data and related meta-data.

```
metadata_file = 'MetaData_IBD.xlsx'; % define the metadata file
xcms_file = 'xcms_IBD_ESIn.csv'; % define the xcms file (ESI-)
S0 = xcms2strc(xcms_file,metadata_file); % creates an structure with the same info as t
```

Data preprocessing (ESI-)

Metabolite annotation

Run metabolite annotation using the MSMS spectral library, the MSMS spectral data, and the XCMS information (m/z, m/z min-max values, RT, RT min-max values, pcgroup). There are different options that can be tuned depending on the quality of the experimental MSMS data and the spectral library used. In this case, we used the following parameters:

```
options_search.type_msms = {'all'};
options_search.collision_energy = {0,'>='};
options_search.list_adducts = {'ESI-'};
options_search.delta_rt = 0.15;
options_search.delta_mz = {20,'ppm'};
options_search.rt_weight = 1;
options_search.mz_weight = 1;
options_search.min_int = {[0.05 250],'%max-absolute'};
options_search.norm_msms = 'base peak';
options_search.isotop_13C = {[0.005,1],'n'};
options_search.min_ions_msms = [10 4];
options_search.min_ions_match = 4;
options_search.dotproduct = {0.7,'A&R','geomean'};
options_search.m = 1.2;
options_search.n = 0.9;
options_search.nr_ids = 1;
options_search.label_pcgroups = 'y';
```

```
options_search.match_msms = {'both'};
options_search.RTs = [];

results_id = lcms_annotate([S0.mz S0.rt S0.min_rt_xcms S0.max_rt_xcms S0.camera{(1:end)
    ms2_exp,ms2_library_esin,ms2_library_esin_info,options_search); % run lcms_annotate
```

Add annotations find using LipiDex to the results_id structure

This function will match the xcms features with the metabolites annotated using LipiDex.

```
[results_id,table_lipids_id] = ...
    lipidex_2_strct(S0.mz,S0.rt,S0.camera{:,3},Folder.ms2_exp,0.15,...
    {20,'ppm'},[],[0 0 -1],results_id);
```

Create a field with IDs from both types of annotation

By running two different (but similar) annotation algorithms using different databases, we increase the potential number of annotated metabolites. Here, if a feature has been annotated using both, the 'HMDB/LipidBlast' and the 'Lipidex' libraries, the first one is kept. Again, a new field in the structure is created (results_id.dbs_name) with the information from both annotations.

```
[~, h] = find(~isnan(results_id.mean_dp(1,:))>0);
[~, l] = find(~isnan(results_id.lipidex_dp(1,:))>0);
hs = zeros(size(results_id.dp,2),1); hs(h) = 2;
ls = zeros(size(results_id.dp,2),1); ls(l) = 1;
ids = hs+ls; % ids: 3:id@lipidex & hmdb;  2:id@hmdb; 1:id@lipidex; 0:no id
results_id.dbs_name = results_id.db_ID_name(1,:); % keep the best match
for j = 1:size(results_id.dp,2)
    if ids(j)==1 % if ID only @lipidex
        results_id.dbs_name{1,j} = results_id.lipidex_ID_name{1,j};
    else
    end
end
```

Create a data set object (PLSToolbox)

This step is optional (you can keep using a data structure if the PLSToolbox is not available), but it is recommended. This function adds ID & classes etc info to a unique dso (**X0**)

```
X0 = annotation2dso(S0,results_id);
```

Within-batch effect correction and data clean-up

This step includes a within-batch effect correction using QCs and the QC-SVRC algorithm, followed by the identification and elimination of unreliable features (RSD(qc)>20%, or detected in blanks).

```
%% qc-svrc
qc = find(strcmp(X0.classid{1,1},'QC')>0);
min_nr_qcs = 9;
lod = -10;
epsilon_range = [2:0.5:5];
gamma_range = 10.^[0:5];
```

```
C_interval = 90;
kfoldcv = numel(qc);
klfolcv_iterations = 1;
[Rtemp] = qcsvrc_batch_2pc(X0.data,qc,min_nr_qcs,lod,...
    epsilon_range,gamma_range,C_interval,kfoldcv,klfolcv_iterations,'subtract','n');
X1 = X0;
X1.data = Rtemp.Xc;
skipped_svr = find(isnan(Rtemp.nSVs_opt));
if numel(skipped_svr)>0
    X1(:,skipped_svr) = []; % exclude features for wich qc-svrc could not be performed
else
end
results_id1 = results_id;
fn = fieldnames(results_id1);
for i = 1:numel(fn)
    results_id1.(fn{i})(:,skipped_svr) = [];
end
```

A new data set object (**X1**) is created with the corrected data, and the *results_id* is also modified accordingly to create the *results_id1*.

Then, exclude from the analysis those features with RSD(QCs)>=20 after QC-SVRC. A new data set object (**X2**) is created with the corrected data, and the *results_id1* is also modified accordingly to create the *results_id2*.

```
clear  Rtemp skipped_svr min_nr_qcs lod epsilon_range gamma_range ...
    C_interval kfoldcv klfolcv_iterations i fn

% rsd @qcs
k = 100*std(X1.data(qc,:))./mean(X1.data(qc,:));
k = find(k<20);
X2 = X1(:,k);
results_id2 = results_id1;
fn = fieldnames(results_id2);
for i = 1:numel(fn)
    results_id2.(fn{i}) = results_id2.(fn{i})(:,k);
end
clear k
```

Finally, the 'background/carry-over' features are excluded from the analysis. A new data set object (**X3**) is created with the corrected data, and the *results_id1* is also modified accordingly to create the *results_id3*.

```
% Blank clean-up
qc = find(strcmp(X2.classid{1,1},'QC')>0);
blank = find(strcmp(X2.classid{1,1},'BLANK')>0);
r = 6;
k = find(median(X2.data(qc,:)) >= median(X2.data(blank,:))*r);
X3 = X2(:,k);
results_id3 = results_id2;
fn = fieldnames(results_id3);
for i = 1:numel(fn)
```

```
    results_id3.(fn{i}) = results_id3.(fn{i})(:,k);
end
clear k r fn i blank
```

Define a new dataset object including only the annotated features:

```
annotated_features = find(~cellfun(@isempty,results_id3.dbs_name(1,:)));
Xa = X3(:,annotated_features);
fn = fieldnames(results_id3);
results_xa = results_id3;
for i = 1:numel(fn)
    results_xa.(fn{i}) = results_id3.(fn{i})(:,annotated_features);
end
```

And save the results

```
save ibd_esin.mat S0 X1 X2 X3 Xa results_id results_id1 results_id2 results_id3 results
```

Create an ESI+/- data set

Create a single ESI+/- dataset with the annotated features.

```
clear
load ibd_esip.mat results_xa Xa
P = Xa;
results_idp = results_xa;
clear Xa results_xa

load ibd_esin.mat results_xa Xa
N = Xa;
results_idn = results_xa;
clear Xa results_xa

%% exclude samples classified as outliers by the clinic.
r = find(strcmp(P.classid{1,1},'EXCLUDE')>0);
N(r,:) = [];
P(r,:) = [];
clear r
% create a single data structure for ESI+ & ESI- annotations
fn = fieldnames(results_idp);
for i = 1:numel(fn)
    results_id.(fn{i}) = [];
    results_id.(fn{i}) = [results_idp.(fn{i}), results_idn.(fn{i})];
end
clear i fn

% Joint N & P data set
for j = 1:size(P,2)
    p{j} = char('positive');
end
P.classname{2,3} = 'mode';
P.classid{2,3} = p;
```

```
for j = 1:size(N,2)
    n{j} = char('negative');
end
N.classname{2,3} = 'mode';
N.classid{2,3} = n;
Xtemp = [P N];
%%
X = dataset(Xtemp.data);
X.labelname{1,1} = Xtemp.labelname{1,1}; X.label{1,1} = Xtemp.label{1,1}; % Sample
X.labelname{1,2} = Xtemp.labelname{1,2}; X.label{1,2} = Xtemp.label{1,2}; % Raw file
X.axisscalename{1,1} = Xtemp.axisscalename{1,1}; X.axisscale{1,1} = Xtemp.axisscale{1,1
X.axisscalename{2,1} = Xtemp.axisscalename{2,1}; X.axisscale{2,1} = Xtemp.axisscale{2,1
X.axisscalename{2,2} = Xtemp.axisscalename{2,2}; X.axisscale{2,2} = Xtemp.axisscale{2,2
X.classname{1,1} = Xtemp.classname{1,1}; X.class{1,1} = Xtemp.classid{1,1}; % F, noF, Q
X.classname{1,2} = Xtemp.classname{1,2}; X.class{1,2} = Xtemp.classid{1,2}; % F-CD, F-U
X.labelname{2,1} = Xtemp.labelname{2,6}; X.label{2,1} = Xtemp.label{2,6}; % db_name
X.classname{2,3} = 'mode'; X.classid{2,3} = Xtemp.classid{2,3}; % positive, negative
```

```
t = readtable('MetaboliteIDs_IBD.xlsx'); % this list was manually revised
X.classname{2,1} = 'class'; X.classid{2,1} = t{:,5};
X.classname{2,2} = 'subclass'; X.classid{2,2} = t{:,6};

results_id.db_ID_class = t{:,5};
results_id.db_ID_subclass = t{:,6};

save ibd_esipn.mat X results_id
```

Data overview

Figure 1 summarizes the main lipid subclasses of the features annotated in the ESI+ and ESI- data sets after data pre-processing and clean-up. The classes with the largest numbers of annotated lipids were glycerophosphocholines, sphingolipids (SLs), glycerophospholipids, and glycerophosphoethanolamines with phosphatidylcholines (PCs), sphingomyelins (SMs), ceramides, lysophosphatidylcholines (LysoPCs), plasmany and plasmenyl PCs, phosphatidylethanolamines (PEs) accounting for 77% of the 952 annotated LC-MS features (483 and 469 measured by ESI+ and ESI-, respectively).

```
% load the ESI+ & ESI- dataset objects, and create a single DSO (ESI+/-)
clear
path1 = '/Volumes/Macintosh HD/Users/2017 LEITAT/Research3/2020 IBD_Lipids/Manuscript_1
Folder.matlab = strcat(path1,'matlab');
cd(Folder.matlab)
load ibd_esipn.mat X results_id
```

Plot the distribution of classes (or subclasses) in the m/z-RT space. For example: distribution of features with sub-classes with at least 15 annotated features.

```
plot_idclasses3dso(X,15,2) % 2:=dbs subclass, 15.=subclases with <15 features (Figure 1
```

Or, we can also plot the distribution observed only in ESI+ data

```
plot_idclasses3dso(X(:,strcmp(X.classid{2,3},'positive')),10,2)
```

Or onlu in ESI- data

```
plot_idclasses3dso(X(:,strcmp(X.classid{2,3},'negative')),10,2)
```

PCA

Principal Component Analysis (PCA) was used for an initial explorative analysis of trends in the data set. **Figure 2** shows the score plots of a two components PCA model explaining 35% of all variation. The random distribution of PC1 and PC2 scores of QC replicates as a function of the injection order, and the tight clustering of the QCs in the PC1-PC2 score depicted in **Figure 2** supported the instrumental stability throughout the analysis. PCA score plots showed a high overlap across the fatigue and non-fatigued patients independently of the type of IBD, and also between IBD patients with and without fatigue, indicating that the neither type of disease nor the presence of fatigue were among the main sources of variance in the data.

```
%% PCA
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
qc = find(strcmp(X.classid{1,1},'QC')>0);
options_pca = pca('options');
options_pca.display = 'off';
options_pca.plots = 'final';
options_pca.preprocessing = {preprocess('default', 'autoscale')};
max_pcs = 2;
pca(X([f nof],:),max_pcs,options_pca); % model excluding QCs & blanks
pca(X([qc f nof],:),max_pcs,options_pca); % model including QCs.
% The injection order is included in the dataset object, so it can be
% directly plotted the variation in the PC scores-space ~run order
```

ASCA

ANOVA Simultaneous Component Analysis (ASCA) was then used to quantify the effects of the type of disease (i.e. UC/CD), fatigue (Yes/No) and their interaction on the metabolic profiles. ASCA provides a multivariate ANOVA by applying a Simultaneous Component Analysis to each of the effects modeled by an ANOVA27. In this study, the ANOVA model included 2-way interactions of two factors: **X** = **Mean** + **Xdisease**+ **Xfatigue** +**Xdisease-fatigue** + **E**. Results summarized in Table 2 revealed that the dominant part of the variation was unrelated to the two considered factors or their interaction. However, results showed small contributions of fatigue (p-value = 0.06) and ‘disease’ (p-value = 0.2), and no effect for the interaction between disease and fatigue. PC scores of the ASCA factor ‘fatigue’ (i.e. **Xfatigue**) are depicted in Figure 3. In this model, the effect associated to fatigue, previously masked by other sources of variability (e.g. between-individual variation) in the initial PCA could be observed.

```
fcd = find(strcmp(X.classid{1,2},'F-CD')>0);
nofcd = find(strcmp(X.classid{1,2},'NoF-CD')>0);
nofuc = find(strcmp(X.classid{1,2},'NoF-UC')>0);
fuc = find(strcmp(X.classid{1,2},'F-UC')>0);
X = X([fcd fuc nofcd nofuc],:); % re-order for better visualization
```

```
cd_ = [find(strcmp(X.classid{1,2},'NoF-CD')>0) find(strcmp(X.classid{1,2},'F-CD')>0)];
uc_ = [find(strcmp(X.classid{1,2},'NoF-UC')>0) find(strcmp(X.classid{1,2},'F-UC')>0)];
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

F = zeros(size(X,1),2);
F(cd_,1) = 1;
F(uc_,1) = 2;

F(f,2) = 1;
F(nof,2) = 2;

F = dataset(F);
F.label{2,1} = {'Disease','Fatigue'};
F.classname{1,1} = 'Disease';
for i = 1:size(X,1)
    if F.data(i,1) == 1
        l{i} = 'CD';
    else
        l{i} = 'UC';
    end
end
F.classid{1,1} = l;
for i = 1:size(X,1)
    if F.data(i,2) == 1
        l{i} = 'Fatigue';
    else
        l{i} = 'No fatigue';
    end
end
F.classid{1,2} = l;
F.classname{1,2} = 'Fatigue';

%% Specify options
opts               = asca('options');
pp                 = preprocess('default', 'autoscale');
opts.preprocessing = pp;
opts.npermutations = 1000;
opts.interactions  = 2;
ncomp              = 6;

% run ASCA
model = asca(X, F, ncomp, opts)
% or type 'asca' in the command window
% and use the following inputs:
% Response: X
% DOE: F
% X-preprocessing: autoscale
% Number of permutations: 1000
% Interactions: 2-way interactions
% remove center points: on
```

Supervised analysis: PLS-DA and t-tests

PLS-DA double cross-validation (2CV)

Supervised PLS-DA was carried out for the assessment of the class separation between fatigued and non-fatigued groups of IBD patients, and for the identification of a metabolic phenotype associated to fatigue. A double cross validation (2CV) strategy was selected for model development and for the assessment of its generalization accuracy. In 2CV, a randomly selected subset of samples is set aside by *k*-fold cross-validation (*k*-fold CV, *k*=9 in this study), and used as a validation set. The remaining samples are then again split into train and test subset by *leave one out*-fold CV for the optimization of the PLS-DA classifier used to predict the test samples. The procedure is repeated until all samples have been included once in the validation set and then, an estimate of the discrimination between classes is calculated using the set of predictions. This way, samples used for the evaluation of the model performance are excluded from model development. The procedure is repeated a number of times (9 in this study) to average the effect of the initial random *k*-fold CV on the results.

- PLS-DA 2CV and permutation testing is very time consuming and th*is step migh take a few hours (depending on the computer). You can skip this step by loading the* ***ibd_esipn_plsda.mat*** *file*.

```
clear
path1 = '/Volumes/Macintosh HD/Users/2017 LEITAT/Research3/2020 IBD_Lipids/Manuscript_I
Folder.matlab = strcat(path1,'matlab');
Folder.plsda2cv = strcat(path1,'BDA_PLSDA_2CV');
addpath(Folder.plsda2cv)
cd(Folder.matlab)
load ibd_esipn
```

Using real-class labels:

```
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
X = X([nof f],:);
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

% 2cv PLSDA method
X = X.data;
I = size(X,1);
y = -ones(I,1);
y(f) = 1; % y=-1, no fatigue; y=+1, fatigue

% Number of permutations
testset = [1:5; 6:10; 11:15; 16:20; 21:25; 26:30; 31:35; 36:40; ...
    41:45; 46 47 nan nan nan];
Perm_real = 9;
Perm = 250;
y = (y+1)/2;

for i = 1:20
    P = randperm(numel(testset));
    testset = reshape(testset(P),size(testset,1),size(testset,2));
    [nmc_2cv(i),Q2_2cv(i),RP,AUROC(i),Y_test(:,i),B,Bfinal(i,:),Tfinal] = ...
        PLSDA_2cv(X,y,6,testset,1,1:I);
```

```
end
Y_test = Y_test*2-1;
```

and then, run a permutation test using randomly asigned class labels:

```
for i = 1:Perm
    IP = randperm(numel(y));
    yp = y(IP);
    disp(i);
    for j = 1:Perm_real
        P = randperm(numel(testset));
        testset = reshape(testset(P),size(testset,1),size(testset,2));
        [nmc_2cvP(i,j),Q2_2cvP(i,j),RP,AUROCP(i,j),YtestP,B,BfinalP(i,j,:)] = PLSDA_2cv
    end
end
perm_test_p_value.NMC = (1+numel(find(mean(nmc_2cvP,2)<mean(nmc_2cv))))/size(nmc_2cvP,1
perm_test_p_value.Q2 = (1+numel(find(mean(Q2_2cv,2)>mean(Q2_2cvP))))/size(nmc_2cvP,1);
perm_test_p_value.AUROC = (1+numel(find(mean(AUROCP,2)>mean(AUROC))))/size(nmc_2cvP,1);
rmpath(Folder.plsda2cv)
```

```
save ibd_esipn_plsda.mat AUROC AUROCP B Bfinal BfinalP nmc_2cv nmc_2cvP Perm_real ...
    Q2_2cv Q2_2cvP testset X y Y_test YtestP
```

PLS-DA feature selection

PLS-DA feature selection using results from the 2CV-permutation testing. First, select the threshold providing the higher classification accuracy.

```
%% Part 1) cut-off selection
clear
path1 = '/Volumes/Macintosh HD/Users/2017 LEITAT/Research3/2020 IBD_Lipids/Manuscript_I
Folder.matlab = strcat(path1,'matlab');
Folder.plsda2cv = strcat(path1,'BDA_PLSDA_2CV');
addpath(Folder.plsda2cv)
cd(Folder.matlab)
load ibd_esipn X
load ibd_esipn_plsda.mat X Bfinal BfinalP Perm_real testset y
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
X = X([nof f],:);
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

alpha =[0.005:0.005:0.1];
breal = Bfinal;
m_breal = mean(breal);
bperm = squeeze(mean(BfinalP,2));

for j = 1:size(m_breal,2)
    switch sign(m_breal(j))
        case 1
            p(j) = numel(find(bperm(:,j)>m_breal(j)))/numel(bperm(:,j));
```

```
        otherwise
            p(j) = numel(find(bperm(:,j)<m_breal(j)))/numel(bperm(:,j));
    end
end
% evaluate CVs @ confidence level
n_keep = [];
for j = 1:numel(alpha)
    n = numel(find(p<=alpha(j)));
    n_keep = [n_keep; n];
end

for j = 1:numel(alpha)
    disp(j)
    [~, keep] = find(p<=alpha(j)); % p<alpha --> comparables
    if numel(keep)>6
        for i = 1:Perm_real
            P = randperm(numel(testset));
            testset = reshape(testset(P),size(testset,1),size(testset,2));
            [nmc_2cv_(i),Q2_2cv_(i),~,AUROC_(i),~,~,~,~]=PLSDA_2cv(X.data(:,keep),y,6,t
        end
        results.alpha(j) = alpha(j);
        results.J(j) = numel(keep);
        results.nmc_2cv(j) = mean(nmc_2cv_);
        results.AUROC(j) = mean(AUROC_);
        results.nmc_2cv_std(j) = std(nmc_2cv_);
        results.AUROC_std(j) = std(AUROC_);
    else
        results.alpha(j) = alpha(j);
        results.J(j) = numel(keep);
        results.nmc_2cv(j) = nan;
        results.AUROC(j) = nan
        results.nmc_2cv_std(j) = nan;
        results.AUROC_std(j) = nan
    end
end
```

Plot the results

```
figure ('name','errorbar select alha BMKs','position',[100,100,450,250])
subplot(1,3,1)
plot(results.alpha,results.J,'ko--','MarkerFaceColor',[0.85 0.33 0.1],'MarkerSize',6)
ylabel('#Retained LCMS features','FontSize',16)
xlabel('\alpha','FontSize',16)
set(gca,'FontSize',14);
box on
xlim([min(results.alpha) max(results.alpha)])
subplot(1,3,2)
errorbar(results.alpha,100*(size(X,1)-results.nmc_2cv)/size(X,1),results.nmc_2cv_std,'k
ylabel('Classification 2CV accuracy','FontSize',16)
xlabel('\alpha','FontSize',16)
set(gca,'FontSize',14);
box on
xlim([min(results.alpha) max(results.alpha)])
subplot(1,3,3)
```

```
errorbar(results.alpha,results.AUROC,results.AUROC_std,'ko--','MarkerFaceColor',[0.85 0
ylabel('2CV AUROC','FontSize',16)
xlabel('\alpha','FontSize',16)
set(gca,'FontSize',14);
box on
xlim([min(results.alpha) max(results.alpha)])
```

Then, using the selected threshold (0.02 in this case), select the features with p-values<=0.02 (*keep_plsda*).

```
% Feature selection. Part 2) feature selection @cutoff= 0.02
load ibd_esipn.mat X
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
clear f nof

breal = Bfinal;
m_breal = mean(breal);
bperm = squeeze(mean(BfinalP,2));
for j = 1:size(m_breal,2)
    switch sign(m_breal(j))
        case 1
            p(j) = numel(find(bperm(:,j)>m_breal(j)))/numel(bperm(:,j));
        otherwise
            p(j) = numel(find(bperm(:,j)<m_breal(j)))/numel(bperm(:,j));
    end
end
[~, keep_plsda] = find(p<=0.02); % p<alpha --> comparables
```

Univariate feature selection

Univariate feature selection using repeated t-test testing and a p-value<0.05 as threshold.

```
load ibd_esipn.mat X
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

% t-test p-values
for j = 1:size(X,2)
    [~, p(j)] = ttest2(X.data(f,j),X.data(nof,j),'VarType','unequal');
end
% fold change
for j = 1:size(X,2)
    fc(j) = log2(mean(X.data(f,j))/mean(X.data(nof,j)));
end
keep_ttest = find(p<0.05);

k = unique([keep_plsda keep_ttest]);
kint = intersect(keep_plsda,keep_ttest);
X.axisscale{2,6} = p;
```

save the index of the select features: (opt. step)

```
save select_bmks.mat keep_ttest keep_plsda
```

Summary of retained features

The following table(s) (Table 4) summarizes the features retained by PLS-DA.

```
clear
load select_bmks.mat keep_plsda
load ibd_esipn.mat X
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
X = X([nof f],keep_plsda);
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

I = size(X,1);
y = -ones(I,1);
    y(f) = 1; % y=-1, no fatigue; y=+1, fatigue

options_plsda = pls('options');
options_plsda.display = 'off';
options_plsda.plots = 'none';
options_plsda.preprocessing = {preprocess('default', 'autoscale')};
options_plsda.orthogonalize = 'off';
max_lvs = 3;
% pls model calc.
model1_train = plsda(X,y,max_lvs,options_plsda);
% regression vector
b = model1_train.reg(:,2);

t1 = X.classid{2,2}'; % subclasses
for i = 1:size(t1,1)
    t2(i,1) = string(t1(i,1));
end
unique_class = unique(t2,'rows');
for i = 1:size(unique_class,1)
    n_unique_class(i,1) = sum(strcmp(t2,unique_class(i)));
end
classes = table(unique_class,n_unique_class);
[classes idx] = sortrows(classes,2,'descend');
unique_class = unique_class(idx);
for i = 1:size(unique_class,1) % # features of each subclass
    [temp_class, ~] = find(strcmp(t2,unique_class(i))>0);
    n_class(i,1) = numel(temp_class);
    n_b_up(i,1) = numel(find(b(temp_class)>0));
    n_b_down(i,1) = numel(find(b(temp_class)<0));
end
% output table
MSs_plsda = table(unique_class,n_class,...
    n_b_up,n_b_down);
[MSs_plsda ~] = sortrows(MSs_plsda,2,'descend')
```

... and the following table summarizes the features retained by t-tests.

```
clear
load select_bmks.mat keep_ttest
load ibd_esipn.mat X
X = X(:,keep_ttest);
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);

% fold change
for j = 1:size(X,2)
   fct(j) = log2(mean(X.data(f,j))/mean(X.data(nof,j)));
end

t1 = X.classid{2,2}'; % subclasses
for i = 1:size(t1,1)
    t2(i,1) = string(t1(i,1));
end
unique_class = unique(t2,'rows');
for i = 1:size(unique_class,1)
    n_unique_class(i,1) = sum(strcmp(t2,unique_class(i)));
end
classes = table(unique_class,n_unique_class);
[classes idx] = sortrows(classes,2,'descend');
unique_class = unique_class(idx);

% # features of each class with r coef > threshold
for i = 1:size(unique_class,1)
[temp_class, ~] = find(strcmp(t2,unique_class(i))>0);
n_class(i,1) = numel(temp_class);
n_b_up(i,1) = numel(find(fct(temp_class)>0));
n_b_down(i,1) = numel(find(fct(temp_class)<0));
end
% output table
MSs_ttest = table(unique_class,n_class,...
    n_b_up,n_b_down);
[MSs_ttest ~] = sortrows(MSs_ttest,2,'descend')
```

Network analysis

In order to extract more information, the Prize-collecting Steiner forest algorithm for Integrative Analysis of Untargeted Metabolomics (PIUMet) algorithm was used for the analysis of the set of 225 annotated features classified as discriminant by at least one of the abovementioned strategie (i.e. k = unique([keep_plsda keep_ttest)).

Network analysis was carried out using PIUMet (http://fraenkel-nsf.csbi.mit.edu/piumet2/). To prepare a suitable input file:

```
clear
load select_bmks.mat keep_ttest keep_plsda
load ibd_esipn.mat X
f = find(strcmp(X.classid{1,1},'F')>0);
nof = find(strcmp(X.classid{1,1},'noF')>0);
```

```
X = X(:,unique([keep_plsda keep_ttest]));
% calc. t-test p-values (this step was already carried out in a prev. section)
for j = 1:size(X,2)
    [~, p(j)] = ttest2(X.data(f,j),X.data(nof,j),'VarType','unequal');
end
% ratio fatigue/no-fatigue
for j = 1:size(X,2)
   ratio_f_nof(j) = median(X.data(f,j))./median(X.data(nof,j));
end
p = -log(p);
piumet_input = table(X.axisscale{2,1}',X.classid{2,3}',p',ratio_f_nof'); % PIUmet input
```

Go to http://fraenkel-nsf.csbi.mit.edu/piumet2/

@Metabolomic Peaks: upload the piumet_input (terminal) file **OR** paste in the box. This is a tab delimited format with three columns: m/z value, the mode of LC ('positive' or 'negative'), and a prize. Select the following parameters: prize function: -log(p-value), number of trees: 10, edge reliability: 2; negative prize degree: 0.0005; number of repeats: 3.

- **PIUMET reference:** Leila Pirhaji, Pamela Milani, Mathias Leidl, Timothy Curran, Julian Avila-Pacheco, Clary B. Clish, Forest White, Alan Saghatalian, Ernest Fraenkel. Revealing Disease-Modifying Pathways by Network Integration of Untargeted Metabolomics. *Nature Methods*, 13,770-776(2016) https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5209295/

The output from PIUMET can be found in the folder PIUMET_results, including the network of protein-protein and protein-metabolite interactions inferred by PIUMet (the network including the metabolite peaks and names of the hidden proteins and metabolites as .html (the files are also available at the Zenodo repository (zenodo.org/deposit/3906482)).

If you have any question, please do not hesitate to contact us. And, of course, we would greatly appreciate your feedback about any bugs you find while using the scripts.

e.mail: guira@uv.es