A Structure Database and *In Silico* Spectral Library for Comprehensive Suspect Screening of Per- and Polyfluoroalkyl Substances (PFASs) in Environmental Media by High-resolution Mass Spectrometry

Supporting Information

Gordon J. Getzinger (ORCID: 0000-0002-5628-1425)[1]*‡, Christopher P. Higgins[2] and P. Lee Ferguson[1]*

Address correspondence to: ggetzinger@exponent.com and lee.ferguson@duke.edu

Department of Civil and Environmental Engineering and Nicholas School of the Environment, Duke University, 121 Hudson Hall, Box 90287, Durham, NC USA

Department of Civil and Environmental Engineering, Colorado School of Mines, 1500 Illinois St., Golden, CO USA

Table of Contents

Database scheme

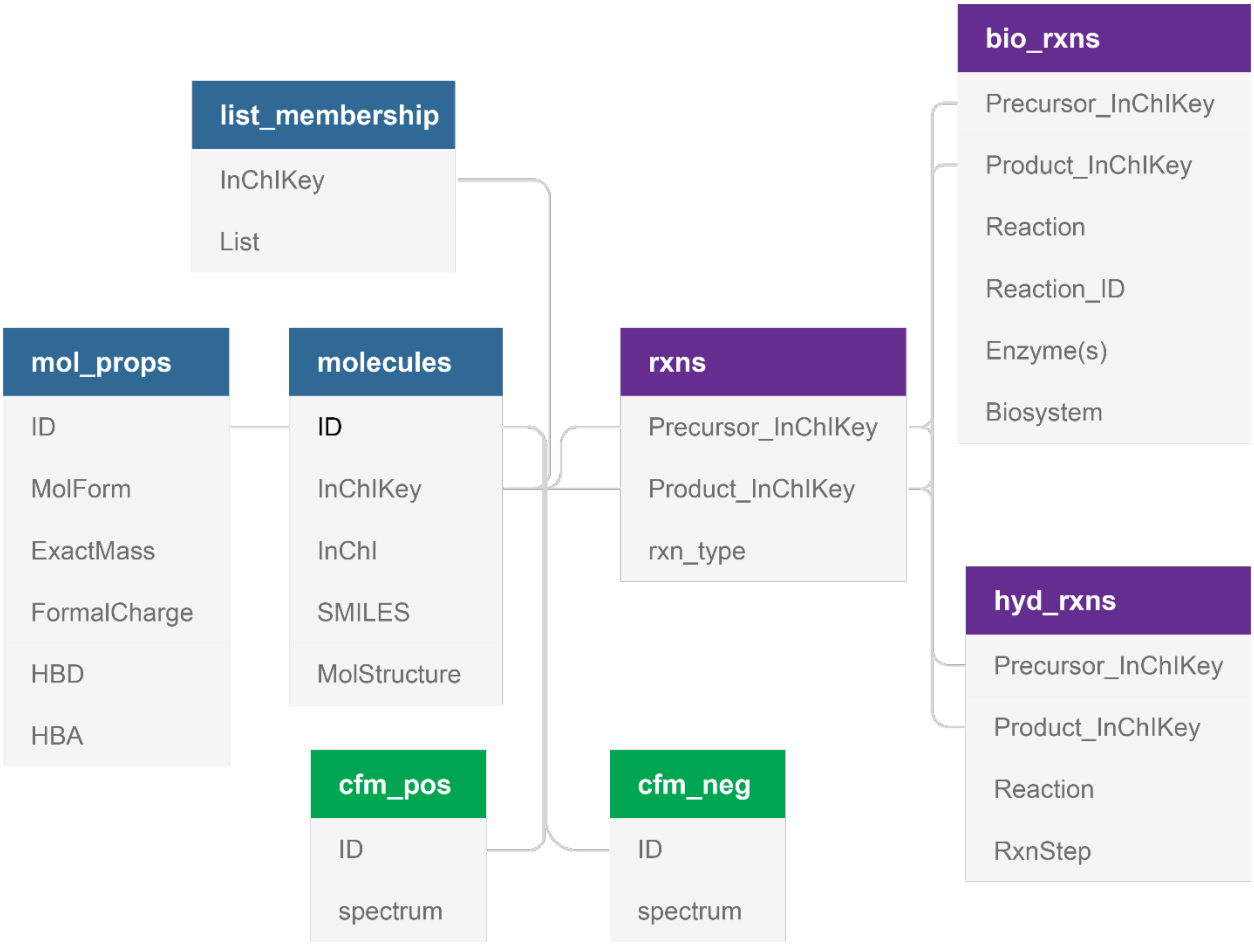

Figure S 1 Database diagram showing database tables, field names in each table, and associated foreign keys. Tables with blue headers indicate those storing molecular identifiers and structural representations of all molecules in the database. Parent-product relationships are stored in reaction tables denoted by purple headings. Tables with green headers indicate those storing negative or positive ionization MS/MS spectra predicted by CFM.

Installation and setup

```
library(devtools)
install_github('gjgetzinger/PFAScreeneR')
install_github('gjgetzinger/cfmR')
library(PFAScreeneR)
library(reticulate)
library(DBI)
library(tidyverse)
library(cfmR)
library(MSnbase)
library(pbapply)
```

Making the molecular database and spectral library

Read input molecules, consolidate structures and predict transformations

Input molecules are read from xls files stored in the system files of the `PFAScreeneR` packge.

```
mollist_path <- system.file(package = 'PFAScreeneR') %>%
  list.dirs() %>%
  .[grep('mass_lists', .)] %>%
  paste0(., '/')
```

The functions for collecting, standardizing and prediting transformations of input molecules are performed by calling `make_pfas_masslist`, which is imported from a python script stored in the system files of this package.

```
py_file <- system.file(package = 'PFAScreeneR') %>%
  list.dirs() %>%
  .[grep('python',.)] %>%
  list.files(full.names = T)
reticulate::source_python(file = py_file)
```

Biotransformation is performed using the `BioTransformer` algorithm. See http://biotransformer.ca for installation intructions and to cite the original work. The path to the `BioTransformer` jar file must be designated.

```
biotrans_jarfile <- '~/path/to/biotransformerjar/biotransformer-1.1.2.jar'
```

Results are written to SDF and SQLite files.

```
result_db <- '/path/to/result/PFAScreenR.db'
```

Calling `make_pfas_masslist` performs all the steps for intial database creation.

```
make_pfas_masslist(
  mollist_path = mollist_path,
  biotrans_jarfile = biotrans_jarfile,
  hyd_sdf = 'hydrolysis_out.sdf',
  biotrans_in_sdf = 'biotrans_in.sdf',
  biotrans_out_sdf = 'biotrans_out.sdf',
  result_sdf = 'PFAScreeneR.sdf',
  result_db = result_db
)
```

Predict MS/MS spectra using CFM

This step requires execution on a Linux compute cluster running a SLURM scheduler and with CFM-ID installed as a commandline tool. See https://cfmid.wishartlab.com for installation instructions for CFM-ID.

```
conn <- dbConnect(RSQLite::SQLite(), result_db)
mass_list <- tbl(conn, 'molecules')
```

Negative ion

```
cfm_predict_batch(
  id = pull(mass_list, ID),
  input_smiles_or_inchi = pull(mass_list, InChI),
  prob_thresh_for_prune = 0.001,
  include_annotations = 1,
  apply_post_processing = 1,
  out_dir = '/path/to/negative_ion/storage/',
  param_filename = '/path/to/negative_ion/param_output1.log',
  config_filename = 'path/to/negative_in/param_config_neg.txt',
  slurm_options = list(partition = 'your_partition', 'mem' = '16G'),
  cpus_per_node = 1,
  nodes = 1000
)
```

Positive ion

```
cfm_predict_batch(
  id = pull(mass_list, ID),
  input_smiles_or_inchi = pull(mass_list, InChI),
  prob_thresh_for_prune = 0.001,
  include_annotations = 1,
  apply_post_processing = 1,
  out_dir = '/path/to/positive_ion/storage/',
  param_filename = '/path/to/positive_ion/param_output1.log',
  config_filename = '/path/to/positive_ion/param_config_pos.txt',
  slurm_options = list(partition = 'your_partition', 'mem' = '16G'),
  cpus_per_node = 1,
  nodes = 1000
)
```

Disconnect from the result database before attempting to retrieve the results.

```
dbDisconnect(conn)
```

Collecting results

Once CFM prediction has completed, results can be written to the databse file.

```
cfm_read_batch(out_dir = '/path/to/negative_ion/storage/',
               table_name = 'cfm_neg',
               output_path = result_db)

cfm_read_batch(out_dir = '/path/to/positive_ion/storage/',
               table_name = 'cfm_pos',
               output_path = result_db)
```

Searching the spectral library

MS/MS spectra in a MGF format can be searched against the spectral library.

```
pfas_exp_msms <-
  readMgfData(list.files(
    system.file(package = 'PFAScreeneR'),
    pattern = '.mgf',
    full.names = T
  ))
```

```r
# get the ionization polarity
pol <-
  ifelse(fData(pfas_exp_msms)$CHARGE == '1-', 'negative', 'positive')
# set the database table name based on ionization polarity
spec_table_name <- ifelse(pol == 'negative', 'cfm_neg', 'cfm_pos')
ID <- as.character(fData(pfas_exp_msms)$ID)
instrument_type <- as.character(fData(pfas_exp_msms)$INSTRUMENT)
ppm <- ifelse(instrument_type == 'Orbitrap', 10, 50)
mol_form <- as.character(fData(pfas_exp_msms)$FORMULA)
exact_mass <-
  as.numeric(as.character(fData(pfas_exp_msms)$EXACT_MASS))
nom_mass_ppm <-
  sapply(exact_mass, function(x)
    1e6 * (diff(x + c(-0.5, 0.5)) / x))
```

Search by ID

Searching by identifier retrieves library spectra from molecules with the same ID. In this case, the ID assigned to experimental spectra (`pfas_exp_msms`) matches the identifiers used to store molecules in the library. The ID in thise case is the first-14 characters of the hashed InChI string (i.e., InChI Key).

```r
sim_id_search <- pbapply::pblapply(seq(along = pfas_exp_msms),
                                   function(x) {
                                     cfm_search(
                                       query_spectrum = pfas_exp_msms[[x]],
                                       spec_table_name = spec_table_name[x],
                                       pol = pol[x],
                                       db_file = db_file,
                                       mol_table_name = 'molecules',
                                       molprop_table_name = 'mol_props',
                                       fun = 'dotproduct',
                                       mol_form = NULL,
                                       exact_mass = NULL,
                                       bin = 0.005,
                                       ID = ID[x],
                                       ppm = ppm[x]
                                     )
                                   })
```

Search by molecular formula

Searhing by molecular formula retrieves only library spectra with a particular molecular formula.

```r
sim_molform_search <- pbapply::pblapply(seq(along = pfas_exp_msms),
                                        function(x) {
                                          cfm_search(
                                            query_spectrum = pfas_exp_msms[[x]],
                                            spec_table_name = spec_table_name[x],
                                            pol = pol[x],
                                            db_file = db_file,
                                            mol_table_name = 'molecules',
                                            molprop_table_name = 'mol_props',
                                            fun = 'dotproduct',
                                            mol_form = mol_form[x],
                                            exact_mass = NULL,
```

```
                                        bin = 0.005,
                                        ID = NULL,
                                        ppm = ppm[x]
                                      )
                                    })
```

Search by neutral exact mass

Searching by neutral exact mass retrieves library spectra with molecular weights within the designated ppm range.

```
sim_neutralmass_search <-
  pbapply::pblapply(seq(along = pfas_exp_msms),
                    function(x) {
                      cfm_search(
                        query_spectrum = pfas_exp_msms[[x]],
                        spec_table_name = spec_table_name[x],
                        pol = pol[x],
                        db_file = db_file,
                        mol_table_name = 'molecules',
                        molprop_table_name = 'mol_props',
                        fun = 'dotproduct',
                        mol_form = NULL,
                        exact_mass = exact_mass[x],
                        bin = 0.005,
                        ID = NULL,
                        ppm = ppm[x]
                      )
                    })
```

Search by neutral nominal mass

Same as above, except with nominal mass.

```
sim_nominalneutralmass_search <- pbapply::pblapply(seq(along = pfas_exp_msms),
                                                   function(x) {
                                                     cfm_search(
                                                       query_spectrum = pfas_exp_msms[[x]],
                                                       spec_table_name = spec_table_name[x],
                                                       pol = pol[x],
                                                       db_file = db_file,
                                                       mol_table_name = 'molecules',
                                                       molprop_table_name = 'mol_props',
                                                       fun = 'dotproduct',
                                                       mol_form = NULL,
                                                       exact_mass = exact_mass[x],
                                                       bin = 0.005,
                                                       ID = NULL,
                                                       ppm = nom_mass_ppm[x]
                                                     )
                                                   })
```

Search by exact precursor m/z

Searching by precursor m/z takes the precursor ion m/z and calculates a series of netural masses based on common ESI/APCI adduct ions. The resultant neutral masses are used as described above for neutral mass look-up.

```
sim_precurmz_search <- pbapply::pblapply(seq(along = pfas_exp_msms),
                                         function(x) {
                                           cfm_search(
                                             query_spectrum = pfas_exp_msms[[x]],
                                             spec_table_name = spec_table_name[x],
                                             pol = pol[x],
                                             db_file = db_file,
                                             mol_table_name = 'molecules',
                                             molprop_table_name = 'mol_props',
                                             fun = 'dotproduct',
                                             mol_form = NULL,
                                             exact_mass = NULL,
                                             bin = 0.005,
                                             ID = NULL,
                                             ppm = ppm[x]
                                           )
                                         })
```

Search by nominal precursor m/z

Same as above, except with nominal mass.

```
sim_nominalprecurmz_search <-
  pbapply::pblapply(seq(along = pfas_exp_msms),
                    function(x) {
                      cfm_search(
                        query_spectrum = pfas_exp_msms[[x]],
                        spec_table_name = spec_table_name[x],
                        pol = pol[x],
                        db_file = db_file,
                        mol_table_name = 'molecules',
                        molprop_table_name = 'mol_props',
                        fun = 'dotproduct',
                        mol_form = NULL,
                        exact_mass = NULL,
                        bin = 0.005,
                        ID = NULL,
                        ppm = nom_mass_ppm[x]
                      )
                    })
```