toffee Documentation

Release latest

ProCan Software Engineering at Children's Medical Research Inst

Apr 16, 2019

Contents

CHAPTER 1

OpenMSToffee: C++

Contents

TODO...

1.1 OpenSwathWorkflow

```
$ OpenSwathWorkflow --helphelp
OpenSwathWorkflow -- Complete workflow to run OpenSWATH
Version: 2.3.0 Jun 21 2018, 07:51:05, Revision: 763e76a
To cite OpenMS:
  Rost HL, Sachsenberg T, Aiche S, Bielow C et al.. OpenMS: a flexible open-source software platform for mass spectrometry data analysis. Nat Meth. 2016; 13, 9: 741-748. doi:10.1038/nmeth.3959.

Usage:
  OpenSwathWorkflow <options>

Options (mandatory options marked with '*'):
  -in <files>* Input files separated by blank (valid formats: 'mzML', 'mzXML', 'sqMass')
  -tr <file>* Transition file ('TraML','tsv','pqp') (valid formats: 'traML', 'tsv', 'pqp')
  -tr_type <type> Input file type -- default: determined from file extension or content
```

(continues on next page)

(continued from previous page)

```
14                                                                                                                    ␣
↪(valid: 'traML', 'tsv', 'pqp')
15    -tr_irt <file>                                                                                                  ␣
↪Transition file ('TraML') (valid formats: 'traML')
16    -rt_norm <file>                                                                                                 RT␣
↪normalization file (how to map the RTs of this run to the ones stored in the␣
↪library). If set, tr_irt may be omitted. (valid formats: 'trafoXML')
17    -swath_windows_file <file>                                                                                      ␣
↪Optional, tab separated file containing the SWATH windows for extraction: lower_
↪offset upper_offset \newline 400 425 \newline ... Note that the first line is a␣
↪header and will be skipped.
18    -sort_swath_maps                                                                                                ␣
↪Sort input SWATH files when matching to SWATH windows from swath_windows_file
19    -use_ms1_traces                                                                                                 ␣
↪Extract the precursor ion trace(s) and use for scoring
20    -enable_uis_scoring                                                                                             ␣
↪Enable additional scoring of identification assays
21    -out_features <file>                                                                                            ␣
↪Output file (valid formats: 'featureXML')
22    -out_tsv <file>                                                                                                 ␣
↪TSV output file (mProphet compatible TSV file) (valid formats: 'tsv')
23    -out_osw <file>                                                                                                 ␣
↪OSW output file (PyProphet compatible SQLite file) (valid formats: 'osw')
24    -out_chrom <file>                                                                                               ␣
↪Also output all computed chromatograms output in mzML (chrom.mzML) or sqMass␣
↪(SQLite format) (valid formats: 'mzML', 'sqMass')
25    -min_upper_edge_dist <double>                                                                                   ␣
↪Minimal distance to the edge to still consider a precursor, in Thomson (default: '0
↪')
26    -rt_extraction_window <double>                                                                                  ␣
↪Only extract RT around this value (-1 means extract over the whole range, a value␣
↪of 600 means to extract around +/- 300 s of the expected elution). (default: '600')
27    -extra_rt_extraction_window <double>                                                                            ␣
↪Output an XIC with a RT-window that by this much larger (e.g. to visually inspect a␣
↪larger area of the chromatogram) (default: '0' min: '0')
28    -mz_extraction_window <double>                                                                                  ␣
↪Extraction window used (in Thomson, to use ppm see -ppm flag) (default: '0.05' min:
↪'0')
29    -ppm                                                                                                            M/
↪z extraction_window is in ppm
30    -sonar                                                                                                          ␣
↪Data is scanning SWATH data
31    -min_rsq <double>                                                                                               ␣
↪Minimum r-squared of RT peptides regression (default: '0.95')
32    -min_coverage <double>                                                                                          ␣
↪Minimum relative amount of RT peptides to keep (default: '0.6')
33    -split_file_input                                                                                               ␣
↪The input files each contain one single SWATH (alternatively: all SWATH are in␣
↪separate files)
34    -use_elution_model_score                                                                                        ␣
↪Turn on elution model score (EMG fit to peak)
35    -readOptions <name>                                                                                             ␣
↪Whether to run OpenSWATH directly on the input data, cache data to disk first or to␣
↪perform a datareduction step first. If you choose cache, make sure to also set␣
↪tempDirectory (default: 'normal' valid: 'nor
36                                                                                                                    mal
↪', 'cache', 'cacheWorkingInMemory', 'workingInMemory')
```

(continues on next page)

(continued from previous page)

```
   -mz_correction_function <name>
↪Use the retention time normalization peptide MS2 masses to perform a mass␣
↪correction (linear, weighted by intensity linear or quadratic) of all spectra.␣
↪(default: 'none' valid: 'none', 'unweighted_regression
                                                                              ',
↪'weighted_regression', 'quadratic_regression', 'weighted_quadratic_regression',
↪'weighted_quadratic_regression_delta_ppm', 'quadratic_regression_delta_ppm')
   -irt_mz_extraction_window <double>
↪Extraction window used for iRT and m/z correction (in Thomson, use ppm use -ppm␣
↪flag) (default: '0.05')
   -ppm_irtwindow
↪IRT m/z extraction_window is in ppm
   -tempDirectory <tmp>
↪Temporary directory to store cached files for example (default: '/tmp/')
   -extraction_function <name>
↪Function used to extract the signal (default: 'tophat' valid: 'tophat', 'bartlett')
   -batchSize <number>
↪The batch size of chromatograms to process (0 means to only have one batch,␣
↪sensible values are around 500-1000) (default: '0' min: '0')

Common UTIL options:
   -ini <file>
↪Use the given TOPP INI file
   -log <file>
↪Name of log file (created only when specified)
   -instance <n>
↪Instance number for the TOPP INI file (default: '1')
   -debug <n>
↪Sets the debug level (default: '0')
   -threads <n>
↪Sets the number of threads allowed to be used by the TOPP tool (default: '1')
   -write_ini <file>
↪Writes the default configuration file
   -write_ctd <out_dir>
↪Writes the common tool description file(s) (Toolname(s).ctd) to <out_dir>
   -no_progress
↪Disables progress logging to command line
   -force
↪Overwrite tool specific checks.
   -test
↪Enables the test mode (needed for internal use only)
   --help
↪Shows options
   --helphelp
↪Shows all options (including advanced)
   --log_arguments
↪Print out all the command line arguments

Debugging:
   -Debugging:irt_trafo <text>
↪Transformation file for RT transform

Parameters for the RTNormalization for iRT petides. This specifies how the RT␣
↪alignment is performed and how outlier detection is applied. Outlier detection can␣
↪be done iteratively (by default) which removes one outlier per iteration or using␣
↪the RANSAC algorithm.:
   -RTNormalization:alignmentMethod <choice>
↪How to perform the alignment to the normalized RT space using anchor points. 'linear
↪': perform linear regression (for few anchor points). 'interpolated': Interpolate␣
↪between anchor points (for few, noise-free
```

(continues on next page)

(continued from previous page)

```
                                                                                e␣
↪anchor points). 'lowess' Use local regression (for many, noisy anchor points). 'b_
↪spline' use b splines for smoothing. (default: 'linear' valid: 'linear',
↪'interpolated', 'lowess', 'b_spline')
  -RTNormalization:outlierMethod <choice>                                       ␣
↪Which outlier detection method to use (valid: 'iter_residual', 'iter_jackknife',
↪'ransac', 'none'). Iterative methods remove one outlier at a time. Jackknife␣
↪approach optimizes for maximum r-squared improvem

↪ent while 'iter_residual' removes the datapoint with the largest residual error␣
↪(removal by residual is computationally cheaper, use this with lots of peptides).␣
↪(default: 'iter_residual' valid: 'iter_residu
                                                                               al
↪', 'iter_jackknife', 'ransac', 'none')
  -RTNormalization:useIterativeChauvenet                                        ␣
↪Whether to use Chauvenet's criterion when using iterative methods. This should be␣
↪used if the algorithm removes too many datapoints but it may lead to true outliers␣
↪being retained.
  -RTNormalization:RANSACMaxIterations <number>                                 ␣
↪Maximum iterations for the RANSAC outlier detection algorithm. (default: '1000')
  -RTNormalization:RANSACMaxPercentRTThreshold <number>                         ␣
↪Maximum threshold in RT dimension for the RANSAC outlier detection algorithm (in␣
↪percent of the total gradient). Default is set to 3% which is around +/- 4 minutes␣
↪on a 120 gradient. (default: '3')
  -RTNormalization:RANSACSamplingSize <number>                                  ␣
↪Sampling size of data points per iteration for the RANSAC outlier detection␣
↪algorithm. (default: '10')
  -RTNormalization:estimateBestPeptides                                         ␣
↪Whether the algorithms should try to choose the best peptides based on their peak␣
↪shape for normalization. Use this option you do not expect all your peptides to be␣
↪detected in a sample and too many 'bad'

↪peptides enter the outlier removal step (e.g. due to them being endogenous peptides␣
↪or using a less curated list of peptides).
  -RTNormalization:InitialQualityCutoff <value>                                 ␣
↪The initial overall quality cutoff for a peak to be scored (range ca. -2 to 2)␣
↪(default: '0.5')
  -RTNormalization:OverallQualityCutoff <value>                                 ␣
↪The overall quality cutoff for a peak to go into the retention time estimation␣
↪(range ca. 0 to 10) (default: '5.5')
  -RTNormalization:NrRTBins <number>                                            ␣
↪Number of RT bins to use to compute coverage. This option should be used to ensure␣
↪that there is a complete coverage of the RT space (this should detect cases where␣
↪only a part of the RT gradient is actually

↪covered by normalization peptides) (default: '10')
  -RTNormalization:MinPeptidesPerBin <number>                                   ␣
↪Minimal number of peptides that are required for a bin to counted as 'covered'␣
↪(default: '1')
  -RTNormalization:MinBinsFilled <number>                                       ␣
↪Minimal number of bins required to be covered (default: '8')

RTNormalization:lowess:
  -RTNormalization:lowess:span <value>                                          ␣
↪Span parameter for lowess (default: '0.666666666666667' min: '0' max: '1')

RTNormalization:b_spline:
```

(continues on next page)

(continued from previous page)

```
  -RTNormalization:b_spline:num_nodes <number>
↪Number of nodes for b spline (default: '5' min: '0')

Scoring parameters section:
  -Scoring:stop_report_after_feature <number>
↪Stop reporting after feature (ordered by quality; -1 means do not stop). (default:
↪'-1')
  -Scoring:rt_normalization_factor <value>
↪The normalized RT is expected to be between 0 and 1. If your normalized RT has a
↪different range, pass this here (e.g. it goes from 0 to 100, set this value to 100)
↪(default: '100')
  -Scoring:quantification_cutoff <value>
↪Cutoff in m/z below which peaks should not be used for quantification any more
↪(default: '0' min: '0')
  -Scoring:write_convex_hull
↪Whether to write out all points of all features into the featureXML
  -Scoring:uis_threshold_sn <number>                                        S/
↪N threshold to consider identification transition (set to -1 to consider all)
↪(default: '0')
  -Scoring:uis_threshold_peak_area <number>
↪Peak area threshold to consider identification transition (set to -1 to consider
↪all) (default: '0')
  -Scoring:scoring_model <choice>
↪Scoring model to use (default: 'default' valid: 'default', 'single_transition')

Scoring:TransitionGroupPicker:
  -Scoring:TransitionGroupPicker:stop_after_feature <number>
↪Stop finding after feature (ordered by intensity; -1 means do not stop). (default:
↪'-1')
  -Scoring:TransitionGroupPicker:min_peak_width <value>
↪Minimal peak width (s), discard all peaks below this value (-1 means no action).
↪(default: '14')
  -Scoring:TransitionGroupPicker:peak_integration <choice>
↪Calculate the peak area and height either the smoothed or the raw chromatogram data.
↪ (default: 'original' valid: 'original', 'smoothed')
  -Scoring:TransitionGroupPicker:background_subtraction <choice>
↪Remove background from peak signal using estimated noise levels. The 'original'
↪method is only provided for historical purposes, please use the 'exact' method and
↪set parameters using the PeakIntegrator:

↪settings. The same original or smoothed chromatogram specified by peak_integration
↪will be used for background estimation. (default: 'none' valid: 'none', 'original',
↪'exact')
  -Scoring:TransitionGroupPicker:recalculate_peaks <choice>
↪Tries to get better peak picking by looking at peak consistency of all picked peaks.
↪ Tries to use the consensus (median) peak border if theof variation within the
↪picked peaks is too large. (default: 'true'

↪valid: 'true', 'false')
  -Scoring:TransitionGroupPicker:use_precursors
↪Use precursor chromatogram for peak picking
  -Scoring:TransitionGroupPicker:recalculate_peaks_max_z <value>
↪Determines the maximal Z-Score (difference measured in standard deviations) that is
↪considered too large for peak boundaries. If the Z-Score is above this value, the
↪median is used for peak boundaries (defau

                                                                          lt
↪value 1.0). (default: '0.75')
```

(continues on next page)

(continued from previous page)

```
108   -Scoring:TransitionGroupPicker:minimal_quality <value>
↪Only if compute_peak_quality is set, this parameter will not consider peaks below␣
↪this quality threshold (default: '-1.5')
109   -Scoring:TransitionGroupPicker:resample_boundary <value>
↪For computing peak quality, how many extra seconds should be sample left and right␣
↪of the actual peak (default: '15')
110   -Scoring:TransitionGroupPicker:compute_peak_quality <choice>
↪Tries to compute a quality value for each peakgroup and detect outlier transitions.␣
↪The resulting score is centered around zero and values above 0 are generally good␣
↪and below -1 or -2 are usually bad. (defa
111
↪ult: 'true' valid: 'true', 'false')
112   -Scoring:TransitionGroupPicker:compute_peak_shape_metrics
↪Calulates various peak shape metrics (e.g., tailing) that can be used for␣
↪downstream QC/QA.
113   -Scoring:TransitionGroupPicker:boundary_selection_method <choice>
↪Method to use when selecting the best boundaries for peaks. (default: 'largest'␣
↪valid: 'largest', 'widest')
114
115  Scoring:TransitionGroupPicker:PeakPickerMRM:
116   -Scoring:TransitionGroupPicker:PeakPickerMRM:sgolay_frame_length <number>
↪The number of subsequent data points used for smoothing.
117
↪This number has to be uneven. If it is not, 1 will be added. (default: '11')
118   -Scoring:TransitionGroupPicker:PeakPickerMRM:sgolay_polynomial_order <number>
↪Order of the polynomial that is fitted. (default: '3')
119   -Scoring:TransitionGroupPicker:PeakPickerMRM:gauss_width <value>
↪Gaussian width in seconds, estimated peak size. (default: '30')
120   -Scoring:TransitionGroupPicker:PeakPickerMRM:use_gauss <choice>
↪Use Gaussian filter for smoothing (alternative is Savitzky-Golay filter) (default:
↪'false' valid: 'false', 'true')
121   -Scoring:TransitionGroupPicker:PeakPickerMRM:peak_width <value>
↪Force a certain minimal peak_width on the data (e.g. extend the peak at least by␣
↪this amount on both sides) in seconds. -1 turns this feature off. (default: '-1')
122   -Scoring:TransitionGroupPicker:PeakPickerMRM:signal_to_noise <value>
↪Signal-to-noise threshold at which a peak will not be extended any more. Note that␣
↪setting this too high (e.g. 1.0) can lead to peaks whose flanks are not fully␣
↪captured. (default: '0.1' min: '0')
123   -Scoring:TransitionGroupPicker:PeakPickerMRM:write_sn_log_messages
↪Write out log messages of the signal-to-noise estimator in case of sparse windows␣
↪or median in rightmost histogram bin
124   -Scoring:TransitionGroupPicker:PeakPickerMRM:remove_overlapping_peaks <choice>
↪Try to remove overlapping peaks during peak picking (default: 'true' valid: 'false',
↪ 'true')
125   -Scoring:TransitionGroupPicker:PeakPickerMRM:method <choice>
↪Which method to choose for chromatographic peak-picking (OpenSWATH legacy on raw␣
↪data, corrected picking on smoothed chromatogram or Crawdad on smoothed␣
↪chromatogram). (default: 'corrected' valid: 'legacy',
126
↪'corrected', 'crawdad')
127
128  Scoring:TransitionGroupPicker:PeakIntegrator:
129   -Scoring:TransitionGroupPicker:PeakIntegrator:integration_type <choice>
↪The integration technique to use in integratePeak() and estimateBackground() which␣
↪uses either the summed intensity, integration by Simpson's rule or trapezoidal␣
↪integration. (default: 'intensity_sum' valid:
130
↪'intensity_sum', 'simpson', 'trapezoid')
```

(continues on next page)

(continued from previous page)

```
  -Scoring:TransitionGroupPicker:PeakIntegrator:baseline_type <choice>
The baseline type to use in estimateBackground() based on the peak boundaries. A rectangular baseline shape is computed based either on the minimal intensity of the peak boundaries, the maximum intensity or

the average intensity (base_to_base). (default: 'base_to_base' valid: 'base_to_base', 'vertical_division', 'vertical_division_min', 'vertical_division_max')

Scoring:DIAScoring:
  -Scoring:DIAScoring:dia_extraction_window <value>
DIA extraction window in Th or ppm. (default: '0.05' min: '0')
  -Scoring:DIAScoring:dia_extraction_unit <choice>
DIA extraction window unit (default: 'Th' valid: 'Th', 'ppm')
  -Scoring:DIAScoring:dia_centroided
Use centroided DIA data.
  -Scoring:DIAScoring:dia_byseries_intensity_min <value>
DIA b/y series minimum intensity to consider. (default: '300' min: '0')
  -Scoring:DIAScoring:dia_byseries_ppm_diff <value>
DIA b/y series minimal difference in ppm to consider. (default: '10' min: '0')
  -Scoring:DIAScoring:dia_nr_isotopes <number>
DIA number of isotopes to consider. (default: '4' min: '0')
  -Scoring:DIAScoring:dia_nr_charges <number>
DIA number of charges to consider. (default: '4' min: '0')
  -Scoring:DIAScoring:peak_before_mono_max_ppm_diff <value>
DIA maximal difference in ppm to count a peak at lower m/z when searching for evidence that a peak might not be monoisotopic. (default: '20' min: '0')

Scoring:EMGScoring:
  -Scoring:EMGScoring:max_iteration <number>
Maximum number of iterations using by Levenberg-Marquardt algorithm. (default: '10')

Scoring:Scores:
  -Scoring:Scores:use_shape_score <choice>
Use the shape score (this score measures the similarity in shape of the transitions using a cross-correlation) (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_coelution_score <choice>
Use the coelution score (this score measures the similarity in coelution of the transitions using a cross-correlation) (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_rt_score <choice>
Use the retention time score (this score measure the difference in retention time) (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_library_score <choice>
Use the library score (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_intensity_score <choice>
Use the intensity score (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_nr_peaks_score <choice>
Use the number of peaks score (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_total_xic_score <choice>
Use the total XIC score (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_sn_score <choice>
Use the SN (signal to noise) score (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_dia_scores <choice>
Use the DIA (SWATH) scores. If turned off, will not use fragment ion spectra for scoring. (default: 'true' valid: 'true', 'false')
  -Scoring:Scores:use_ms1_correlation
Use the correlation scores with the MS1 elution profiles
  -Scoring:Scores:use_sonar_scores
Use the scores for SONAR scans (scanning swath)
```

(continues on next page)

(continued from previous page)

```
159   -Scoring:Scores:use_ms1_fullscan
↪Use the full MS1 scan at the peak apex for scoring (ppm accuracy of precursor and
↪isotopic pattern)
160   -Scoring:Scores:use_uis_scores
↪Use UIS scores for peptidoform identification
```

1.2 Internal Class Structure

class OpenMSToffeeWorkflow : **public** TOPPBase

Public Types

enum FileFormat

Values:

TSV

SQLITE

Public Functions

OpenMSToffeeWorkflow(bool *testing* = false)

struct FileArguments

Public Members

std::string **toffeeFilePath**
input tof file path

std::string **srlFilePath**
input srl file path (tsv or pqp)

std::string **alignmentTSVFilePath**
input alightment tsv file path

std::string **outputFilePath**
output file path

std::string **inputRTTrafoXML**
if not empty, use this to specify RT norm (trafoXML)

std::string **outputRTTrafoXML**
if not empty, save RT norm to here (trafoXML)

FileFormat **format**
defines if using TSV or SQLite

1.2.1 Extracting data from *mzML* and *mzXML* files

class HDF5ChromatogramConsumer : **public** IMSDataConsumer

Public Types

using MapType = OpenMS::PeakMap

using SpectrumType = *MapType*::SpectrumType

using ChromatogramType = *MapType*::ChromatogramType

Public Functions

HDF5ChromatogramConsumer(**const** std::string &*h5FilePath*)

~HDF5ChromatogramConsumer()

void **consumeSpectrum**(*SpectrumType* &*s*)

void **consumeChromatogram**(*ChromatogramType* &*c*)

void **setExpectedSize**(size_t *expectedSpectra*, size_t *expectedChromatograms*)

void **setExperimentalSettings**(**const** OpenMS::ExperimentalSettings &*exp*)

Public Static Functions

void **chromatogramToHDF5**(**const** std::string &*mzMLFilePath*, **const** std::string &*h5FilePath*)
Save an mzML chromatogram file to a much more (>100x) compressed HDF5 file This data is saved as a series of 1D vectors:

- **names** gives the transition ids of the chromatograms
- **offset** gives the index into the retention time and intensity vectors for the corresponding transition id
- **size** gives the size of the data in the retention time and intensity vectors for the corresponding transition id. I.e. its data resides at [offset, offset + size)
- **retentionTime** all retention time data concatenated into a single vector
- **intensity** all intensity data concatenated into a single vector

Parameters

- `mzMLFilePath`: path the the mzML file generated by OpenSwath
- `h5FilePath`: output file path

class RTNormalisation
Calculate the world to iRT normalisation transformation using raw SWATH-MS data contained in a toffee file, and a list of precursor and product ions that can be used for alignment

Public Functions

RTNormalisation(**const** std::string &*toffeeFilePath*, **const** std::string &*alignmentTSVFilePath*)

Parameters

- `toffeeFilePath`: the toffee file for which we wish to calculate the world to iRT normalisation

- `alignmentTSVFilePath`: the path to a TSV file of retention time alignment precursor and product ions

void **updateNormalisationParams**(**const** OpenMS::Param &*param*)
Update the default parameters to input into the normalisation algorithms.

void **updateMzCorrectionFunction**(**const** std::string &*mzCorrectionFunction*)
Update the default method for correcting the mass over charge.

void **updateMinRSquared**(double *minRSquared*)
Update the default minimum R-squared value in regression fitting method.

void **updateMinCoverage**(double *minCoverage*)
Update the default minimum coverage of the fit regression method.

void **updateChromExtractParams**(**const** OpenMS::ChromExtractParams &*param*)
Update the chromatogram extraction configuration.

void **updateFeatureFinderParams**(**const** OpenMS::Param &*param*)
Update the feature finding configuration.

OpenMS::TransformationDescription **calculateNormalisation**() **const**
Calculate the world to iRT normalisation.

OpenMS::TransformationDescription **calculateNormalisationFromFile**(**const** std::string &*inputTrafoPath*) **const**
Calculate the world to iRT normalisation.

void **saveToFile**(**const** OpenMS::TransformationDescription &*transform*, **const** std::string &*trafoXMLFilePath*) **const**
Save a previously calculated iRT normalisation to file.

Warning this function does not check if the transformation is in world to iRT coordinates, or its inverse. It relies on the user to take care!

void **calculateAndSaveToFile**(**const** std::string &*trafoXMLFilePath*) **const**
Calculate the world to iRT normalisation and save it to file.

Public Static Functions

OpenMS::Param **defaultNormalisationParams**()
Default parameters to input into the normalisation algorithms. These can be updated, see below.

static std::string **defaultMzCorrectionFunction**()
Default method for correcting the mass over charge. This can be updated, see below.

static double **defaultMinRSquared**()
Default minimum R-squared value in regression fitting method. This can be updated, see below.

static double **defaultMinCoverage**()
Default minimum coverage of the fit regression method. This can be updated, see below.

CHAPTER 2

OpenMSToffee: Python

Contents

2.1 Working with *toffee* files

2.1.1 Calculating RT normalisation

```
$ toffee_openms_rt_normalisation --help
usage: toffee_openms_rt_normalisation [-h]
                                      toffee_filename alignment_filename
                                      transformation_xml_filename

Calculate the retention time normalisation for a toffee file using OpenMS as
the wrapper

positional arguments:
  toffee_filename       The output filename (*.tof)
  alignment_filename    The input alignment library (iRT) filename (*.tsv)
  transformation_xml_filename
                        The output transformation XML file (*.trafoXML)

```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
```

2.2 Converting SRLs

```
$ srl_peakview_to_openms --help
usage: srl_peakview_to_openms [-h] [-output_dir OUTPUT_DIR]
                              [-minimum_number_of_transitions MINIMUM_NUMBER_OF_
↪TRANSITIONS]
                              [-maximum_number_of_transitions MAXIMUM_NUMBER_OF_
↪TRANSITIONS]
                              [-mz_cutoff MZ_CUTOFF]
                              [-precursor_product_limit PRECURSOR_PRODUCT_LIMIT]
                              [--drop_modifications] [--make_pqp]
                              [--keep_intermediate_files] [--debug]
                              sciex_filename

Convert a Sciex ProteinPilot/PeakView/OneOmics SRL into a format that can be
used by OpenMS

positional arguments:
  sciex_filename        The input filename (*.txt)

optional arguments:
  -h, --help            show this help message and exit
  -output_dir OUTPUT_DIR
                        The directory to save the output files. By default it
                        will be the same as the input file.
  -minimum_number_of_transitions MINIMUM_NUMBER_OF_TRANSITIONS
                        Define the minimum number of transitions for any PSM
  -maximum_number_of_transitions MAXIMUM_NUMBER_OF_TRANSITIONS
                        Define the maximum number of transitions for any PSM
  -mz_cutoff MZ_CUTOFF  Filter out any precursor ions with a mass over charge
                        below this threshold
  -precursor_product_limit PRECURSOR_PRODUCT_LIMIT
                        Filter out any product ions with a mass over charge
                        with this many Da of the precursor ion
  --drop_modifications  Remove all modifications from the final SRL
  --make_pqp            Create the new PQP sqlite format in addition the TSV
  --keep_intermediate_files
                        Remove intermediate files that were created
  --debug               Switch on more detailed logging
```

2.3 Internal Class Structure

OpenMSToffee.log.**set_stream_logger**(*name='OpenMSToffee'*, *level=20*, *format_string=None*, *fname=None*)

Add a stream handler for the given name and level to the logging module. By default, this logs all boto3 messages to `stdout`.

```
>>> import OpenMSToffee as omt
>>> omt.log.set_stream_logger(name='OpenMSToffee', level=logging.INFO)
```

Parameters

- **name** (*string*) – Log name
- **level** (*int*) – Logging level, e.g. `logging.INFO`
- **format_string** (*str*) – Log message format

class OpenMSToffee.srl_peakview_to_openms.**OpenSwathLibraryFromPeakview**(*peakview_fname*, *output_basename=None*, *output_dir=''*, *minimum_number_of_transitions*, *maximum_number_of_transitions*, *mz_cutoff=400.0*, *precursor_product_limit=10.0*, *modifications_to_keep='CAM'*, *make_pqp=False*, *clean_up_files=True*, *debug=True*)

Convert SRL file formats between PeakView and OpenMS. Furthermore, run the decoy generation on the OpenMS data once it has been generated

DROP_TEXT = 'DROP'

OPENSWATH_INDEX_COLS = ['TransitionGroupId']

OPENSWATH_SORT_ASCENDING = [True, False]

OPENSWATH_SORT_COLS = ['TransitionGroupId', 'LibraryIntensity']

PEAKVIEW_HEADERS = ['Q1', 'Q3', 'iRT', 'stripped_sequence', 'relative_intensity', 'uni

PEAKVIEW_RTCAL_PROTEIN = '[RT-Cal protein]'

PV_TO_OS_COL_MAPPING = {'Annotation': 'Annotation', 'CE': 'CollisionEnergy', 'Decoy':

convert()

convert_peakview_to_openswath(*df*)
Internal class method, exposed only for testing

classmethod create_calibration(*df_irt*, *basename*)
Internal class method, exposed only for testing

create_decoys(*df_library*, *basename*)
Internal class method, exposed only for testing

filter_and_normalise_openswath(*df*)
Internal class method, exposed only for testing

open_peak_view()
Internal class method, exposed only for testing

remove_minimum_fragments(*df*)
Internal class method, exposed only for testing

rename_modifications(*mod_peptide_col*)
Internal class method, exposed only for testing. Convert the modification format of PeakView into UniMod format

classmethod split_to_srl_and_alignment(*df*)
Internal class method, exposed only for testing

|| Master: || Dev:

`OpenMSToffee` is a wrapper around `OpenSwath` that allows us to use the toffee file format. It is available as a Docker image at `cmriprocan/openms-toffee`.

We follow the OpenVDB style guide for the C++ and PEP-8 for our python code, so please aim to stay consistent with the rest of the code base. Contributions will be pass through peer review and style will be one element that is reviewed.

CHAPTER 3

Changes

CHAPTER 4

Change Log

4.1 0.14

4.1.1 0.14.1

- Changed license to MIT and fixed documentation for https://openms-toffee.readthedocs.io
- Bumped version of `toffee` to 0.13.1, also MIT licensed

4.2 0.13

4.2.1 0.13.12

- Bumped version of `toffee` to 0.12.16, which now includes the mzML to toffee to mzML conversions
- Removed all mzML to toffee conversions from this package and deleted tests

4.2.2 0.13.11

- Bumped version of `toffee` to 0.12.9, which now includes the toffee to mzML conversion (PD-793)

4.2.3 0.13.10

- Bumped version of `toffee` to 0.12.4
- Many fixes to accommodate change in structure of the dia-test-data repository

4.2.4 0.13.9

- Added ability to convert toffee files to Spectra HDF5 files (PD-793)

4.2.5 0.13.8

- Bumped version of `toffee` to 0.11.1

4.2.6 0.13.6

- Bumped version of `toffee` to 0.10.6 (that fixes PD-749) and added unit test

4.2.7 0.13.5

- Bumped version of `toffee` to 0.10.5 (that fixes PD-735)

4.2.8 0.13.4

- Pinned version of `toffee` to 0.10.4 (that fixes PD-727)

4.2.9 0.13.3

- A few bug fixes for the mzML to toffee conversion

4.2.10 0.13.2

- Updated to version 1.2.4 of `cmriprocan/openms`.

4.2.11 0.13.1

- Significant refactor of the `XML` to `toffee` conversion process. This has been pulled apart into smaller modules such that individual parts can be tested. Importantly, the code that converts `mzML` and `mzXML` to `HDF5` is now shared by the process that then converts these to toffee. This improves memory performance, and means that we can test both conversions.

4.3 0.12

4.3.1 0.12.1

- Directly linking to upstream image in Dockerfile: `cmriprocan/openms:1.2.3` which:
  - Updated `OpenMS` to version `CMRI-ProCan-v1.1.2`
  - Updated `PyProphet` to `2.0.2`

4.4 0.11

4.4.1 0.11.1

- Updated `OpenMS` to version `CMRI-ProCan-v1.1.1` with hot-fix to solve https://github.com/OpenMS/OpenMS/issues/3860

4.5 0.10

4.5.1 0.10.1

- Updated `OpenMS` to version `CMRI-ProCan-v1.1.0` that incorporates upstream `Release2.4.0` with our changes that enable toffee files to be used on the command line

4.6 License

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

P

R

S