Package 'IDSL.FSA'

June 29, 2023

Type Package

Title Fragmentation Spectra Analysis (FSA)

Version 1.2

Depends R (>= 4.0)

Suggests readxl

Author Sadjad Fakouri-Baygi [aut] (<https://orcid.org/0000-0002-6864-6911>),
Dinesh Barupal [cre, aut] (<https://orcid.org/0000-0002-9954-8628>)

Maintainer Dinesh Barupal <dinesh.barupal@mssm.edu>

Description The 'IDSL.FSA' package was designed to annotate standard .msp (mass spectra format) and .mgf (Mascot generic format) files using mass spectral entropy similarity, dot product (cosine) similarity, and normalized Euclidean mass error (NEME) followed by intelligent pre-filtering steps for rapid spectra searches. 'IDSL.FSA' also provides a number of modules to convert and manipulate .msp and .mgf files. The 'IDSL.FSA' workflow was integrated in the 'IDSL.CSA' and 'IDSL.NPA' packages introduced in <doi:10.1021/acs.analchem.3c00376>.

License MIT + file LICENSE

URL https://github.com/idslme/idsl.fsa

BugReports https://github.com/idslme/idsl.fsa/issues

Encoding UTF-8

Archs i386, x64

NeedsCompilation no

Repository CRAN

Date/Publication 2023-06-29 14:10:02 UTC

R topics documented:

fragmentation_spectra_annotator
Fragmentation Spectra Annotator

Description

This module annotates fragmentation spectra from .MSP files.

Usage

```
fragmentation_spectra_annotator(path, MSPfile = "", libFSdb,
libFSdbIDlist, targetedPrecursorType = NA, ratio2basePeak4nSpectraMarkers = 0,
allowedNominalMass = FALSE, allowedWeightedSpectralEntropy = TRUE,
noiseRemovalRatio = 0.01, roundingDigitPrefiltering = 1, minMatchedNumPeaks = 1,
massError = 0, maxNEME = 0, minIonRangeDifference = 0, minCosineSimilarity,
minEntropySimilarity, minRatioMatchedNspectraMarkers,
spectralEntropyDeviationPrefiltering, massErrorPrecursor = NA, RTtolerance = NA,
exportSpectraParameters = NULL, number_processing_threads = 1)
```

Arguments

path	Address of .msp file(s)
MSPfile	name of the .msp file
libFSdb	A converted .msp library reference file using the 'msp2FSdb' module which is an FSDB produced by the IDSL.FSA package.
libFSdbIDlist	Ion markers object from the FSDB reference
targetedPrecursorType	
	A vector of targeted precursor types
ratio2basePeak4nSpectraMarkers	
	Ratio of peaks in fragmentation spectra to the basepeak to calculate minimum qualified number of matched abundant peaks
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.
allowedWeightedSpectralEntropy	
	c(TRUE, FALSE). Weighted entropy to transform low abundant signals prior to calculating entropy similarity score. Please see the reference for details on wight transformation.
noiseRemovalRatio	
	noise removal ratio ([0 - 1])relative to the basepeak to measure entropy similarity score.
roundingDigitPrefiltering	
	Level of pre-filtering
minMatchedNumPeaks	
	Minimum matched number of peaks
massError	Mass accuracy in Da
maxNEME	Maximum value for Normalized Euclidean Mass Error (NEME) in mDa
minIonRangeDifference	
	Minimum distance (Da) between lowest and highest matched m/z to prevent matching only isotopic envelopes
minCosineSimilarity	
	Minimum cosine similarity score
minEntropySimilarity	
	Minimum entropy similarity score

minRatioMatchedNspectraMarkers

        Minimum percentage of detection of abundant library peaks in percentage

spectralEntropyDeviationPrefiltering

        Spectral entropy deviation for pre-filtering

massErrorPrecursor

        Mass accuracy (Da) to find precursor m/z in .msp files

RTtolerance   Retention time tolerance (min)

exportSpectraParameters

        Parameters for export MS/MS match figures

number_processing_threads

        Number of processing threads for multi-threaded processing

Value

A dataframe of matched spectra

FSA_aggregate   *aggregation method for FSA*

Description

This module is to optimize the 'indexVec' variable by removing elements that have redundant 'idVec' numbers.

Usage

```
FSA_aggregate(idVec, variableVec, indexVec, targetVar)
```

Arguments

idVec   a vector of id numbers. Repeated id numbers are allowed

variableVec   a vector of variable of the interest such as RT, m/z, etc.

indexVec   a vector of indices

targetVar   the targeted value in 'variableVec'

Value

a clean indexVec after removing redundant 'idVec'.

```
FSA_annotation_text_repel
```
FSA annotation text repel

Description

This function is to set annotations on the spectra plots with a reasonable distance to avoid overlying annotations.

Usage

```
FSA_annotation_text_repel(FSAspectra, nGridX, nGridY)
```

Arguments

FSAspectra	FSAspectra
nGridX	number of grids on the x-axis
nGridY	number of grids on the y-axis

Value

labels

```
FSA_dir.create              FSA_dir.create
```

Description

A module to create directories after removing the existing directory with the same name to prevent data interferences.

Usage

```
FSA_dir.create(folder, allowedUnlink = FALSE)
```

Arguments

folder	folder
allowedUnlink	allowedUnlink

Value

when the original folder was deleted and recreated successfully, 'TRUE' is returned by this function.

FSA_FSdb_xlsxAnalyzer   *FSA FSdb xlsx Analyzer*

Description

This function processes the spreadsheet of the 'FSDB' tab to ensure the parameter inputs are consistent with the requirements of the IDSL.FSA pipeline.

Usage

```
FSA_FSdb_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet   FSA spreadsheet

Value

This function returns the FSDB parameters to feed the 'FSdb_file_generator' function.

FSA_loadRdata   *FSA loadRdata*

Description

This function loads .Rdata files into a variable.

Usage

```
FSA_loadRdata(fileName)
```

Arguments

fileName   is an '.Rdata' file.

Value

The called variable into the new assigned variable name.

FSA_locate_regex   *FSA Locate regex*

Description

Locate indices of the pattern in the string

Usage

```
FSA_locate_regex(string, pattern, ignore.case = FALSE, perl = FALSE, fixed = FALSE,
useBytes = FALSE)
```

Arguments

string	a string as character
pattern	a pattern to screen
ignore.case	ignore.case
perl	perl
fixed	fixed
useBytes	useBytes

Details

This function returns 'NULL' when no matches are detected for the pattern.

Value

A 2-column matrix of location indices. The first and second columns represent start and end positions, respectively.

Examples

```
pattern <- "Cl"
string <- "NaCl.5HCl"
Location_Cl <- FSA_locate_regex(string, pattern)
```

FSA_logRecorder   *FSA logRecorder*

Description

FSA_logRecorder

Usage

```
FSA_logRecorder(messageQuote, allowedPrinting = TRUE)
```

Arguments

messageQuote   messageQuote

allowedPrinting
                   allowedPrinting

Value

a line of communication messages is exported to the console and the log .txt file.

FSA_message   *FSA message*

Description

FSA_message

Usage

```
FSA_message(messageQuote, failedMessage= TRUE)
```

Arguments

messageQuote   messageQuote

failedMessage   failedMessage

Value

a line of communication messages is exported to the console.

FSA_msp2Cytoscape *FSA Cytoscape Files Generator*

Description

This function generates necessary files from pairwise MSP blocks analysis to create Cytoscape networks.

Usage

```
FSA_msp2Cytoscape(path, MSPfile = "", mspVariableVector = NULL,
mspNodeID = NULL, massError = 0.01, RTtolerance = NA, minEntropySimilarity = 0.75,
allowedNominalMass = FALSE, allowedWeightedSpectralEntropy = TRUE,
noiseRemovalRatio = 0.01, number_processing_threads = 1)
```

Arguments

path	address of .msp file or an FSDB
MSPfile	name of .msp file
mspVariableVector	
	a vector of msp variables
mspNodeID	msp Node ID which is the ID that is required for the 'specsim' ID generation
massError	Mass accuracy in Da
RTtolerance	Retention time tolerance (min) to match msp blocks. Select *NA* to ignore retention time match. This option is so helpful to find co-occurring compounds.
minEntropySimilarity	
	Minimum entropy similarity score
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.
allowedWeightedSpectralEntropy	
	c(TRUE, FALSE). Weighted entropy to measure entropy similarity score.
noiseRemovalRatio	
	noise removal ratio relative to the basepeak to measure entropy similarity score (in percent)
number_processing_threads	
	Number of processing threads for multi-threaded processing

Value

node_attributes_dataFrame	
	node_attributes dataframe. A string to store using 'writeTable' function of R after a tab separation.
edge_dataFrame	edge dataframe. A string to store using the 'writeTable' function of R after a tab separation.

correlation_network

> correlation_network dataframe. A string to store using the 'writeTable' function of R after a tab separation.

FSDB   Fragmentation spectra database (FSDB) object

exclusionMSPnoideid

> A vector of MSP node ids which can be excluded to create a library of unique MSP blocks.

filteredNetworkSIF

> A filtered network in the cytoscape SIF format that does not have redundant MSP blocks within a RT window.

References

Shannon, P., Markiel, A., Ozier, O., Baliga, N.S., Wang, J.T., Ramage, D., Amin, N., Schwikowski, B. and Ideker, T., (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11), 2498-2504, doi:10.1101/gr.1239303

Examples

```
path_extdata <- system.file("extdata", package = "IDSL.FSA")
mspFileName <- "Kynurenine_Kynurenic_acid.msp"
##
listCytoscape <- FSA_msp2Cytoscape(path = path_extdata,
MSPfile = mspFileName, mspVariableVector = c("Name", "Collision_energy"),
mspNodeID = NULL, massError = 0.01, RTtolerance = NA, minEntropySimilarity = 0,
noiseRemovalRatio = 0, allowedNominalMass = FALSE,
allowedWeightedSpectralEntropy = TRUE, number_processing_threads = 1)
##
FSDB <- listCytoscape[["FSDB"]]
##
temp_wd <- tempdir() # just a temporary folder to save results
##
write.table(listCytoscape[["node_attributes_dataFrame"]], paste0(temp_wd,
"/node_attributes_dataFrame.txt"), quote = FALSE, sep = "\t", row.names = FALSE,
col.names = FALSE)
##
write.table(listCytoscape[["correlation_network"]], paste0(temp_wd,
"/correlation_network.sif"), quote = FALSE, sep = "\t", row.names = FALSE,
col.names = FALSE)
##
write.table(listCytoscape[["edge_dataFrame"]], paste0(temp_wd,
"/edge_dataFrame.txt"), quote = FALSE, sep = "\t", row.names = FALSE,
col.names = FALSE)
##
```

FSA_msp_annotator   *FSA msp annotator*

Description

This function arranges the parameters for the annotation process

Usage

```
FSA_msp_annotator(PARAM_SPEC, libFSdb, address_input_msp, output_path,
allowedVerbose = TRUE)
```

Arguments

PARAM_SPEC	a parameter driven from the 'FSA_SpectraSimilarity_xlsxAnalyzer' module.
libFSdb	a converted .msp library reference files (FSDB) using the 'msp2FSdb' module
address_input_msp	
	address of the .msp files
output_path	output path
allowedVerbose	c(TRUE, FALSE). A 'TRUE' allowedVerbose provides messages about the flow of the function.

Value

A dataframe of matched annotated spectra stored in the output directory.

FSA_plotFSdb2Spectra   *plot FSdb to Spectra*

Description

plot FSdb to Spectra

Usage

```
FSA_plotFSdb2Spectra(path, allowedUnlink = TRUE, annexName = "", FSdb,
selectedFSdbIDs = NULL, number_processing_threads = 1, allowedVerbose = TRUE)
```

Arguments

`path`	Address of .msp file(s)
`allowedUnlink`	allowedUnlink
`annexName`	annexName
`FSdb`	FSdb
`selectedFSdbIDs`	
	selected FSdb IDs. When 'NULL', the entire FSDB blocks are plotted.
`number_processing_threads`	
	Number of processing threads for multi-threaded processing
`allowedVerbose`	c(TRUE, FALSE). A 'TRUE' allowedVerbose provides messages about the flow of the function.

Value

spectra_figure object

`FSA_R.aggregate`	*aggregate function for IDSL.FSA*

Description

This module ensures that the 'aggregate' function of R returns a list type of data.

Usage

```
FSA_R.aggregate(FSAvec)
```

Arguments

`FSAvec`	a vector of data

Value

listIDFSAvec

```
FSA_SpectraSimilarity_xlsxAnalyzer
```
FSA SpectraSimilarity xlsx Analyzer

Description

This function processes the spreadsheet of the 'SpectraSimilarity' tab to ensure the parameter inputs are consistent with the requirements of the IDSL.FSA pipeline.

Usage

```
FSA_SpectraSimilarity_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet   FSA spreadsheet

Value

This function returns the FSA SpectraSimilarity parameters to feed the 'FSA_msp_annotator' module.

```
FSA_spectra_marker_generator
```
FSA Spectra Marker Generator

Description

This function generates spectra markers

Usage

```
FSA_spectra_marker_generator(FSdb, ratio2basePeak4nSpectraMarkers = 0,
aggregationLevel = NA)
```

Arguments

FSdb   FSdb object from the 'msp2FSdb' module

ratio2basePeak4nSpectraMarkers

     Ratio of peaks in fragmentation spectra to the basepeak to calculate minimum qualified number of matched abundant peaks

aggregationLevel

     c(NA, 0, 1, 2, 3). When 'NA', this function returns a matrix for the spectra markers. When integer numbers are used, the ion marker masses are grouped by a rounding digit equal to this number.

Value

    spectraMarkerMass

                    a grouped or a matrix of ion marker masses corresponding to FSdb ids

    nSpectraMarkers

                    number of spectra markers for each FSdb id

---

    FSA_uniqueMSPblockTagger

                            FSA Unique MSP Block Tagger

Description

This function removes similar MSP blocks. This function aggregates MSP blocks based on the
'Name' values.

Usage

```
FSA_uniqueMSPblockTagger(path, MSPfile = "", aggregateBy = "Name",
massError = 0.01, RTtolerance = NA, minEntropySimilarity = 0.75,
noiseRemovalRatio = 0.01, allowedNominalMass = FALSE,
allowedWeightedSpectralEntropy = TRUE, plotSpectra = FALSE,
number_processing_threads = 1)
```

Arguments

    path   Address of .msp file or an FSDB

    MSPfile   name of .msp file

    aggregateBy   a variable to aggregate the MSP blocks based on

    massError   Mass accuracy in Da

    RTtolerance   Retention time tolerance (min) to match msp blocks. Select *NA* to ignore reten-
                    tion time match.

    minEntropySimilarity

                    Minimum entropy similarity score

    noiseRemovalRatio

                    noise removal ratio relative to the basepeak to measure entropy similarity score
                    (in percent)

    allowedNominalMass

                    c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.

    allowedWeightedSpectralEntropy

                    c(TRUE, FALSE). Weighted entropy to measure entropy similarity score.

    plotSpectra   c(TRUE, FALSE)

    number_processing_threads

                    Number of processing threads for multi-threaded processing

Value

a list of similar MSP blocks is returned at the end and a subsetted .msp and FSDB files are saved in the 'path' directory.

FSA_uniqueMSPblockTaggerUntargeted

FSA_uniqueMSPblockTaggerUntargeted

Description

FSA_uniqueMSPblockTaggerUntargeted

Usage

```
FSA_uniqueMSPblockTaggerUntargeted(path, MSPfile_vector,
minCSAdetectionFrequency = 20, minEntropySimilarity = 0.75, massError = 0.01,
massErrorPrecursor = 0.01, RTtolerance = 0.1, noiseRemovalRatio = 0.01,
allowedNominalMass = FALSE, allowedWeightedSpectralEntropy = TRUE,
plotSpectra = FALSE, number_processing_threads = 1)
```

Arguments

path	Address of .msp file(s)
MSPfile_vector	A vector of names of .msp files or one .msp file name.
minCSAdetectionFrequency	
	minimum CSA detection frequency
minEntropySimilarity	
	minimum EntropySimilarity
massError	Mass accuracy in Da
massErrorPrecursor	
	Mass accuracy (Da) to find precursor m/z in .msp files
RTtolerance	Retention time tolerance (min)
noiseRemovalRatio	
	noise removal ratio ([0 - 1])relative to the basepeak to measure entropy similarity score.
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.
allowedWeightedSpectralEntropy	
	c(TRUE, FALSE). Weighted entropy to transform low abundant signals prior to calculating entropy similarity score. Please see the reference for details on wight transformation.
plotSpectra	c(TRUE, FALSE)
number_processing_threads	
	Number of processing threads for multi-threaded processing

Value

uniqueMSPvariants

FSA_workflow *FSA workflow*

Description

This function executes the FSA workflow.

Usage

```
FSA_workflow(spreadsheet)
```

Arguments

spreadsheet   FSA spreadsheet

Value

This function organizes the FSA file processing for better performance using the template spreadsheet.

FSA_xlsxAnalyzer *FSA xlsx Analyzer*

Description

This function processes the spreadsheet of the FSA parameters to ensure the parameter inputs are consistent with the requirements of the IDSL.FSA pipeline.

Usage

```
FSA_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet   FSA spreadsheet

Value

This function returns the FSA parameters to feed the FSA_workflow function.

FSdb2msp *Fragmentation Spectra DataBase (FSDB) to MSP*

Description

This function converts FSDB R objects into .msp standard files.

Usage

```
FSdb2msp(path, FSdbFileName = "", UnweightMSP = FALSE,
number_processing_threads = 1)
```

Arguments

path   address of .msp file(s)

FSdbFileName   name of the FSDB library name including '.Rdata' extension

UnweightMSP   to unweight fragmentation patterns

number_processing_threads
         Number of processing threads for multi-threaded processing

Value

The .msp file is stored in the same folder

FSdb2PeakXcolSubsetter

FSdb2PeakXcolSubsetter

Description

FSdb2PeakXcolSubsetter

Usage

```
FSdb2PeakXcolSubsetter(FSdb_address, peak_alignment_folder,
metavariable = "idsl.ipa_collective_peakids", number_processing_threads = 1)
```

Arguments

FSdb_address   FSdb_address

peak_alignment_folder
         peak_alignment_folder

metavariable   metavariable

number_processing_threads
         Number of processing threads for multi-threaded processing

Value

peakXcol	peakXcol
peak_height	peak_height
peak_area	peak_area
peak_R13C	peak_R13C

FSdb2precursorType   *Precursor Types from Fragmentation Spectra DataBase (FSDB)*

Description

This function finds potential ionization pathways for molecular formulas using a vector of InChIKey values from an FSDB. This function only searches for the first 14 InChIKey letters; and therefore, may result with multiple potential precursor types.

Usage

```
FSdb2precursorType(InChIKeyVector, libFSdb, tableIndicator = "Frequency",
number_processing_threads = 1)
```

Arguments

InChIKeyVector	A vector of InChIKey values. This value may contain whole InChIKey strings or first 14 InChIKey letters.
libFSdb	A converted MSP library reference file using the 'msp2FSdb' module which is an FSDB produced by the IDSL.FSA package.
tableIndicator	c("Frequency", "PrecursorMZ"). To show frequency or a median of 'PrecursorMZ' values in the output dataframe for each precursor type.
number_processing_threads	
	Number of processing threads for multi-threaded processing

Value

A matrix of frequency for each InChIKey in the FSDB. The matrix column headers represent precursor types.

Examples

```
address_input_msp <- system.file("extdata", package = "IDSL.FSA")
MSPfile_vector <- c("Kynurenine_Kynurenic_acid.msp")
libFSdb <- msp2FSdb(path = address_input_msp, MSPfile_vector)
##
InChIKeyVector <- c("HCZHHEIFKROPDY-UHFFFAOYSA-N", "YGPSJZOEDVAXAB-QMMMGPOBSA-N")
precursor_type_table <- FSdb2precursorType(InChIKeyVector, libFSdb,
tableIndicator = "Frequency", number_processing_threads = 1)
```

FSdb_file_generator *FSdb file generator*

Description

This function generates FSDB objects

Usage

```
FSdb_file_generator(PARAM_FSdb, output_path = NULL)
```

Arguments

PARAM_FSdb 'PARAM_FSdb' parameters obtained by the 'FSA_FSdb_xlsxAnalyzer' function.

output_path output_path

Value

An FSDB object

FSdb_subsetter *FSdb subsetter*

Description

FSdb subsetter

Usage

```
FSdb_subsetter(FSdb, inclusionIDs = NULL, exclusionIDs = NULL)
```

Arguments

FSdb FSdb

inclusionIDs inclusionIDs

exclusionIDs exclusionIDs

Value

subsetted FSdb

---

mgf2msp   *MGF to MSP*

Description

This function converts .mgf (Mascot generic format) files into the .msp (mass spectra) format.

Usage

```
mgf2msp(path, MGFfile = "")
```

Arguments

path   address of the .mgf file.

MGFfile   name of the file with the .mgf extension.

Value

The .msp files are saved in the same location.

Examples

```
temp_wd <- tempdir() # just a temporary folder
path_extdata <- system.file("extdata", package = "IDSL.FSA")
MGFfile <- "Training_000.mgf"
file.copy(from = paste0(path_extdata, "/", MGFfile), to = temp_wd)
mgf2msp(path = temp_wd, MGFfile)
```

---

msp2FSdb   *msp to Fragmentation Spectra DataBase (FSDB)*

Description

This function converts .msp (mass spectra format) files into a readable R object.

Usage

```
msp2FSdb(path, MSPfile_vector = "", massIntegrationWindow = 0,
allowedNominalMass = FALSE, allowedWeightedSpectralEntropy = TRUE,
noiseRemovalRatio = 0.01, number_processing_threads = 1)
```

Arguments

`path`	Address of .msp file(s)
`MSPfile_vector`	A vector of names of .msp files or one .msp file name.
`massIntegrationWindow`	
	Mass window in Da to integrate adjacent peaks in the fragmentation spectra
`allowedNominalMass`	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.
`allowedWeightedSpectralEntropy`	
	c(TRUE, FALSE). Weighted entropy to transform low abundant signals prior to calculating entropy similarity score. Please see the reference for details on wight transformation.
`noiseRemovalRatio`	
	noise removal ratio ([0 - 1])relative to the basepeak to measure entropy similarity score.
`number_processing_threads`	
	Number of processing threads for multi-threaded processing

Value

`logFSdb`	Parameters used to create the FSDB object
`PrecursorMZ`	A vector of precursor m/z values
`Precursor Type`	A vector of precursor adduct types
`Retention Time`	A vector of retention time values
`Num Peaks`	A vector of num peaks values indicating number of ions for each fragment spectra
`Spectral Entropy`	
	A vector of spectral entropy values
`FragmentList`	A list of fragment ions
`MSPLibraryParameters`	
	A dataframe of tabulated headers and their values for each msp block

Note

This function was designed not only to achieve the fastest computational speed; but also can standardize .msp files that were generated by inconsistent settings.

References

Li, Y., Kind, T., Folz, J., Vaniya, A., Mehta, S.S. and Fiehn, O. (2021). Spectral entropy outperforms MS/MS dot product similarity for small-molecule compound identification. *Nature methods*, 18(12), 1524-1531, doi:10.1038/s4159202101331z

Examples

```
path_extdata <- system.file("extdata", package = "IDSL.FSA")
MSPfile <- c("Kynurenine_Kynurenic_acid.msp")
sampleFSdb <- msp2FSdb(path = path_extdata, MSPfile)
```

---

msp2TrainingMatrix   *msp to Fragmentation Spectra DataBase (FSDB)*

Description

This function creates an aligned table from the spectra in the .msp file

Usage

```
msp2TrainingMatrix(path, MSPfile = "", minDetectionFreq = 1,
selectedFSdbIDs = NULL, dimension = "wide", massAccuracy = 0.01,
allowedNominalMass = FALSE, allowedWeightedSpectralEntropy = TRUE,
noiseRemovalRatio = 0.01, number_processing_threads = 1)
```

Arguments

path	Address of .msp file or an FSDB
MSPfile	A .msp file name or FSDB in .Rdata format
minDetectionFreq	
	A minimum detection frequency for an ion across the entire spectra
selectedFSdbIDs	
	selected MSP block/FSDB IDs to limit the screening to specific ion blocks
dimension	c("wide", "long"). *wide* or *long* alignment matrix output
massAccuracy	A mass accuracy (Da)
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.
allowedWeightedSpectralEntropy	
	c(TRUE, FALSE). Weighted entropy to transform low abundant signals prior to calculating entropy similarity score. Please see the reference for details on wight transformation.
noiseRemovalRatio	
	noise removal ratio ([0 - 1])relative to the basepeak to measure entropy similarity score.
number_processing_threads	
	Number of processing threads for multi-threaded processing

Value

A FSDB file (.Rdata) and aligned spectra table (.csv) are stored in the same directory.

Examples

```
temp_wd <- tempdir() # just a temporary folder
path_extdata <- system.file("extdata", package = "IDSL.FSA")
MSPfile <- "Kynurenine_Kynurenic_acid.msp"
file.copy(from = paste0(path_extdata, "/", MSPfile), to = temp_wd)
msp2TrainingMatrix(path = temp_wd, MSPfile, minDetectionFreq = 1)
```

---

mspPosNegSplitter   *MSP Pos/Neg Splitter*

Description

This function separates the positive and negative MSP blocks.

Usage

```
mspPosNegSplitter(path, MSPfile = "", number_processing_threads = 1)
```

Arguments

path   address of the .msp file.

MSPfile   name of the file with the .msp extension.

number_processing_threads
                    Number of processing threads for multi-threaded processing

Value

The .msp files are saved in the same location with '_Neg.msp' and '_Pos.msp' extensions.

Examples

```
temp_wd <- tempdir() # just a temporary folder
path_extdata <- system.file("extdata", package = "IDSL.FSA")
MSPfile <- "Kynurenine_Kynurenic_acid.msp"
file.copy(from = paste0(path_extdata, "/", MSPfile), to = temp_wd)
mspPosNegSplitter(temp_wd, MSPfile)
```

---

plotFSdb2SpectraCore   *plot spectra from FSdb core*

Description

This function plots spectra figures from FSdb objects generated using the 'msp2FSdb' function.

Usage

```
plotFSdb2SpectraCore(FSdb, index)
```

Arguments

FSdb   FSdb

index   index

Value

spectra_figure object

Examples

```
## To create the FSdb object
temp_wd <- tempdir() # just a temporary folder
path_extdata <- system.file("extdata", package = "IDSL.FSA")
MSPfile <- c("Kynurenine_Kynurenic_acid.msp")
file.copy(from = paste0(path_extdata, "/", MSPfile), to = temp_wd)
FSdb <- msp2FSdb(path = temp_wd, MSPfile)
## To plot spectra
index <- 1
plotFSdb2SpectraCore(FSdb, index)
```

---

spectral_entropy_calculator
                              Spectral Entropy Calculator

Description

This module calculates spectral entropy for a fragmentation pattern using a method described by
the reference paper.

Usage

```
spectral_entropy_calculator(FragmentList, allowedWeightedSpectralEntropy = TRUE,
noiseRemovalRatio = 0.01)
```

Arguments

FragmentList   A matrix (m/z, int) of fragmentation pattern after intensity adjustment

allowedWeightedSpectralEntropy

        c(TRUE, FALSE). Weighted entropy to transform low abundant signals prior to calculating entropy similarity score. Please see the reference for details on weight transformation.

noiseRemovalRatio

        noise removal ratio ([0 - 1])relative to the basepeak to measure entropy similarity score.

Value

spectralEntropy

        spectral entropy

NumPeaks   NumPeaks

FragmentList   A matrix of two-columns after intensity normalization relative to summation of intensities AND entropy weight transformation when is selected.

Note

noise removal on intensities should be performed prior to feeding to this function

References

Li, Y., Kind, T., Folz, J., Vaniya, A., Mehta, S.S. and Fiehn, O. (2021). Spectral entropy outperforms MS/MS dot product similarity for small-molecule compound identification. *Nature methods*, 18(12), 1524-1531, doi:10.1038/s4159202101331z

Examples

```
FragmentList <- cbind(seq(50, 600, length.out = 10), seq(10, 90, length.out = 10))
SE <- spectral_entropy_calculator(FragmentList)
print(SE[[1]])
```

---

spectral_entropy_similarity_score
Spectral Entropy Calculator

Description

This module measures similarity of spectral entropies between 'PEAK_A' and 'PEAK_B' fragment spectra using a method described by the reference paper.

Usage

```
spectral_entropy_similarity_score(PEAK_A, S_PEAK_A, PEAK_B, S_PEAK_B, massError,
allowedNominalMass = FALSE)
```

Arguments

PEAK_A	A matrix (m/z, int) of fragmentation spectra
S_PEAK_A	Spectral entropy of PEAK_A
PEAK_B	A matrix (m/z, int) of fragmentation spectra
S_PEAK_B	Spectral entropy of PEAK_B
massError	Mass accuracy in Da
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.

Value

spectral entropy similarity between 0 - 1

References

Li, Y., Kind, T., Folz, J., Vaniya, A., Mehta, S.S. and Fiehn, O. (2021). Spectral entropy outperforms MS/MS dot product similarity for small-molecule compound identification. *Nature methods*, 18(12), 1524-1531, doi:10.1038/s4159202101331z

Examples

```
allowedWeightedSpectralEntropy <- TRUE
##
A <- cbind(seq(50, 160, length.out = 10), seq(10, 90, length.out = 10))
sA <- spectral_entropy_calculator(A, allowedWeightedSpectralEntropy)
S_PEAK_A <- sA[[1]]
PEAK_A <- sA[[3]]
##
B <- cbind(seq(50, 160, length.out = 10), seq(50, 60, length.out = 10))
sB <- spectral_entropy_calculator(A, allowedWeightedSpectralEntropy)
S_PEAK_B <- sB[[1]]
PEAK_B <- sB[[3]]
##
allowedNominalMass = TRUE
entropyScore <- spectral_entropy_similarity_score(PEAK_A, S_PEAK_A, PEAK_B,
S_PEAK_B, allowedNominalMass)
```

---

spectra_1A1B_mixer   *Mixer 1:1 spectra A and B*

Description

This function creates 1:1 mixed AB spectra for spectral entropy calculation

Usage

```
spectra_1A1B_mixer(PEAK_A, PEAK_B, massError = 0, allowedNominalMass = FALSE)
```

Arguments

PEAK_A	A matrix (m/z, int) of fragmentation spectra
PEAK_B	A matrix (m/z, int) of fragmentation spectra
massError	Mass accuracy in Da
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.

Value

A matrix of 1:1 mixing spectra. First and second columns represent intensity-weighted average m/z and cumulated intensity, respectively.

---

spectra_integrator   *Spectra Integrator*

Description

This function integrates individual m/z peaks from multiple chromatogram scans (spectra) into summed m/z peaks using a mass accuracy or nominal masses.

Usage

```
spectra_integrator(stackedSpectra, massError = 0, allowedNominalMass = FALSE)
```

Arguments

stackedSpectra	A matrix of two columns of the stacked spectra. First and second columns should represent m/z and intensity, respectively.
massError	Mass accuracy in Da
allowedNominalMass	
	c(TRUE, FALSE). Select 'TRUE' only for nominal mass analysis.

Value

A matrix of integrated spectra. First and second columns represent intensity-weighted average m/z and cumulated intensity, respectively.

Examples

```
data(stackedSpectra)
massError <- 0.005 # Da
Integrated_spectra <- spectra_integrator(stackedSpectra[, 1:2], massError)
```

---

spectra_ion_filter   *Spectra Ion Filter*

Description

This function can detect m/z peaks that are related to each other across selected spectra lists.

Usage

```
spectra_ion_filter(spectraList, indexSpectraList = length(spectraList), massError,
minPercentageDetectedScans = 10, rsdCutoff = 0, pearsonRHOthreshold = NA)
```

Arguments

spectraList   a list of matrices of m/z and intensity values for each chromatogram scan

indexSpectraList

        a vector of spectra indices for the analysis. This vector should have at least 3
        elements to run this function.

massError   required mass error for m/z values

rsdCutoff   Relative standard deviations (in percent) to remove constant peaks (usually noisy
        peaks)

minPercentageDetectedScans

        Minimum percentage of detected scans for an m/z peak

pearsonRHOthreshold

        A threshold for pairwise Pearson's correlation coefficient across the selected
        spectra lists. This feature is recommended to find co-occurring peaks within a
        chromatographic peak. This feature may be used to eliminate instrument noises
        from MS2 data channels within an MS1 chromatographic peak for DDA analy-
        sis.

Value

A matrix of m/z and cumulated intensities across the 'indexSpectraList' spectra

---

stackedSpectra   *Example for a stacked spectra*

Description

A data to test the 'spectra_integrator' function.

Usage

```
data("stackedSpectra")
```

Format

mz a numeric vector of m/z values

int a numeric vector of intensities

scan_number a numeric vector of chromatogram scan numbers

Details

The 'scan_number' column is not necessary to test the 'spectra_integrator' function.

Examples

```
data(stackedSpectra)
```

UFSA_element_sorter   *Element Sorter*

Description

This function sorts 84 elements in the periodic table for molecular formula deconvolution.

Usage

```
UFSA_element_sorter()
```

Value

A string vector of elements

Examples

```
Elements <- UFSA_element_sorter()
```

UFSA_formula_vector_generator
                    Molecular Formula Vector Generator

Description

This function convert a molecular formulas into a numerical vector

Usage

```
UFSA_formula_vector_generator(molecular_formula, Elements, LElements = length(Elements),
allowedRedundantElements = FALSE)
```

Arguments

`molecular_formula`
                molecular formula

`Elements`   a string vector of elements. This value must be driven from the 'element_sorter' function.

`LElements`   number of elements. To speed up loop calculations, consider calculating the number of elements outside of the loop.

`allowedRedundantElements`
                'TRUE' should be used to deconvolute molecular formulas with redundant elements (e.g. CO2CH3O), and 'FALSE' should be used to skip such complex molecular formulas.(default value)

Value

a numerical vector for the molecular formula. This function returns a vector of -Inf values when the molecular formula has elements not listed in the 'Elements' string vector.

Examples

```
molecular_formula <- "C12H2Br5Cl3O"
Elements <- UFSA_element_sorter()
mol_vec <- UFSA_formula_vector_generator(molecular_formula, Elements)
##
regenerated_molecular_formula <- UFSA_hill_molecular_formula_printer(Elements, mol_vec)
```

`UFSA_hill_molecular_formula_printer`
Print Hill Molecular Formula

Description

This function produces molecular formulas from a list numerical vectors in the Hill notation system

Usage

```
UFSA_hill_molecular_formula_printer(MolVecMat, Elements, LElements = length(Elements))
```

Arguments

`MolVecMat`   A matrix of numerical vectors of molecular formulas in each row.

`Elements`   A vector string of the used elements.

`LElements`   LElements

Value

A vector of molecular formulas

Examples

```
Elements <- c("C", "H", "O", "N", "Br", "Cl")
MoleFormVec1 <- c(2, 6, 1, 0, 0, 0) # C2H6O
MoleFormVec2 <- c(8, 10, 2, 4, 0 ,0) # C8H10N4O2
MoleFormVec3 <- c(12, 2, 1, 0, 5, 3) # C12H2Br5Cl3O
MolVecMat <- rbind(MoleFormVec1, MoleFormVec2, MoleFormVec3)
H_MolF <- UFSA_hill_molecular_formula_printer(MolVecMat, Elements)
```

UFSA_ionization_pathway_deconvoluter

Ionization Pathway Deconvoluter

Description

This function deconvolutes ionization pathways into a coefficient and a numerical vector to simplify prediction ionization pathways.

Usage

```
UFSA_ionization_pathway_deconvoluter(IonPathways, Elements, LElements = length(Elements))
```

Arguments

IonPathways	A vector of ionization pathways. Pathways should be like [Coeff*M+ADD1-DED1+...] where "Coeff" should be an integer between 1-9 and ADD1 and DED1 may be ionization pathways. ex: 'IonPathways <- c("[M]+", "[M+H]+", "[2M-Cl]-", "[3M+CO2-H2O+Na-KO2+HCl-NH4]-")'
Elements	A vector string of the used elements
LElements	Counts of elements

Value

A list of adduct calculation values for each ionization pathway.

Examples

```
Elements <- UFSA_element_sorter()
IonPathways <- c("[M]+", "[M+H]+", "[2M-Cl]-", "[3M+CO2-H2O+Na-KO2+HCl-NH4]-")
Ion_DC <- UFSA_ionization_pathway_deconvoluter(IonPathways, Elements)
```

```
UFSA_precursorType_corrector
```
                              UFA Precursor Type Corrector

Description

Precursor type corrector from MSP files. This function initially attempts to standardize the precursor types to be consistent with the 'ionization_pathway_deconvoluter' module of the IDSL.SUFA package.

Usage

```
UFSA_precursorType_corrector(precursorType, ionMode = NULL)
```

Arguments

precursorType   precursorType

ionMode   ionMode

Value

correctedPrecursorType

Examples

```
uncorrectedPrecursorType <- c("[M]+", "[M+H]+", "[2M-Cl]-", "[3M+COO-H2O+Na-KO2+HCl-NH4]-")
precursorType <- UFSA_precursorType_corrector(uncorrectedPrecursorType, ionMode = NULL)
```

```
xlsx2msp
```
                         xlsx to MSP

Description

This function creates .msp files from an organized spreadsheet of fragmentation data.

Usage

```
xlsx2msp(path, xlsxFileName = "", number_processing_threads = 1)
```

Arguments

path   address of the spreadsheet

xlsxFileName   name of the file with the .xlsx extension.

number_processing_threads
                    Number of processing threads for multi-threaded processing

Value

The .msp files are saved in the same location.

Note

The spreadsheet should have only one column for the following headers (case-sensitive): c('ID', 'mz_fragment', 'int_fragment', 'Name')

Examples

```
temp_wd <- tempdir() # just a temporary folder
path_extdata <- system.file("extdata", package = "IDSL.FSA")
xlsxFileName <- "PFAS_MSe.xlsx"
file.copy(from = paste0(path_extdata, "/", xlsxFileName), to = temp_wd)
xlsx2msp(temp_wd, xlsxFileName)
```

Index