Package 'IDSL.UFA'

May 18, 2023

Type Package

Title United Formula Annotation (UFA) for HRMS Data Processing

Version 2.0

Depends R (>= 4.0)

Imports IDSL.IPA (>= 2.7), readxl

Suggests GA

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 A pipeline to annotate chromatography peaks from the 'IDSL.IPA' workflow <doi:10.1021/acs.jproteome.2c00120> with molecular formulas of a prioritized chemical space using an isotopic profile matching approach. The 'IDSL.UFA' workflow only requires mass spectrometry level 1 (MS1) data for formula annotation. The 'IDSL.UFA' methods was described in <doi:10.1021/acs.analchem.2c00563> .

License MIT + file LICENSE

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

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

Encoding UTF-8

Archs i386, x64

NeedsCompilation no

Repository CRAN

Date/Publication 2023-05-18 21:20:02 UTC

R topics documented:

`aggregatedIPdbListGenerator`
aggregatedIPdbListGenerator

Description

aggregatedIPdbListGenerator

Usage

```
aggregatedIPdbListGenerator(MassMAIso)
```

Arguments

`MassMAIso`	MassMAIso

Value

AggregatedList

aligned_molecular_formula_annotator *Aligned Molecular Formula Annotator*

Description

This function detects frequent molecular formulas across multiple samples on the aligned peak table matrix.

Usage

```
aligned_molecular_formula_annotator(PARAM)
```

Arguments

PARAM	a parameter driven from the UFA_xlsxAnalyzer module.

detect_formula_sets *Organic Class Detection by Repeated Unit Patterns*

Description

This function sorts a vector of molecular formulas to aggregate organic compound classes with repeated/non-repeated substructure units. This function only works for molecular formulas with following elements: c("As", "Br", "Cl", "Na", "Se", "Si", "B", "C", "F", "H", "I", "K", "N", "O", "P", "S")

Usage

```
detect_formula_sets(molecular_formulas, ratio_delta_HBrClFI_C = 2,
mixed.HBrClFI.allowed = FALSE, min_molecular_formula_class = 2,
max_number_formula_class = 100, number_processing_threads = 1)
```

Arguments

molecular_formulas	a vector of molecular formulas
ratio_delta_HBrClFI_C	c(2, 1/2, 0). 2 to detect structures with linear carbon chains such as PFAS, lipids, chlorinated paraffins, etc. 1/2 to detect structures with cyclic chains such as PAHs. 0 to detect molecular formulas with a fixed structures but changing H/Br/Cl/F/I atoms similar to PCBs, PBDEs, etc.
mixed.HBrClFI.allowed	mixed.HBrClFI.allowed = c(TRUE, FALSE). Select 'FALSE' to detect halogenated-saturated compounds similar to PFOS or select 'TRUE' to detect mixed halogenated compounds with hydrogen.

min_molecular_formula_class
: minimum number of molecular formulas in each class. This number should be greater than or equal to 2.

max_number_formula_class
: maximum number of molecular formulas in each class

number_processing_threads
: Number of processing threads for multi-threaded computations.

Value

A matrix of clustered classes of organic molecular formulas.

Examples

```
molecular_formulas <- c("C3F7O3S", "C4F9O3S", "C5F11O3S", "C6F9O3S", "C8F17O3S",
"C9F19O3S", "C10F21O3S", "C7ClF14O4", "C10ClF20O4", "C11ClF22O4", "C11Cl2F21O4",
"C12ClF24O4")
##
ratio_delta_HBrClFI_C <- 2 # to aggregate polymeric classes
mixed.HBrClFI.allowed <- FALSE # To detect only halogen saturated classes
min_molecular_formula_class <- 2
max_number_formula_class <- 20
##
classes <- detect_formula_sets(molecular_formulas, ratio_delta_HBrClFI_C,
mixed.HBrClFI.allowed, min_molecular_formula_class, max_number_formula_class,
number_processing_threads = 1)
```

---

element_sorter *Element Sorter*

Description

This module sorts 84 non-labeled and 14 labeled elements in the periodic table for molecular formula deconvolution and isotopic profile calculation.

Usage

```
element_sorter(ElementList = "all", alphabeticalOrder = TRUE)
```

Arguments

ElementList
: A string vector of elements needed for isotopic profile calculation. The default value for this parameter is a vector string of entire elements.

alphabeticalOrder
: 'TRUE' should be used to sort the elements for elemental deconvolution (default value), 'FALSE' should be used to keep the input order.

Value

Elements	A string vector of elements (alphabetically sorted or unsorted)
massAbundanceList	A list of isotopic mass and abundance of elements.
Valence	A vector of electron valences.

Examples

```
EL_mass_abundance_val <- element_sorter()
```

---

extendedSENIORrule *extended SENIOR rule check*

Description

This function checks whether a molecular formula follows the extended SENIOR rule.

Usage

```
extendedSENIORrule(mol_vec, valence_vec, ionization_correction = 0)
```

Arguments

mol_vec	A vector of the deconvoluted molecular formula
valence_vec	A vector of the valences from the molecular formula. Valences may be acquired from the 'IUPAC_Isotopes' data.
ionization_correction	A number to compensate for the ionization losses/gains. For example, '-1' for [M+H/K/Na] ionization pathways and '+1' for [M-H] ionization pathway.

Value

rule2	TURE for when the molecular formula passes the rule and FALSE for when the molecular formula fails to pass the rule.

formula_adduct_calculator *Formula Adduct Calculator*

Description

This function takes a formula and a vector of ionization pathways and returns the adduct formulas.

Usage

```
formula_adduct_calculator(molecular_formula, IonPathways)
```

Arguments

molecular_formula	molecular formula
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]-")'

Value

A vector of adduct formulas

Examples

```
molecular_formula = "C15H10O7"
IonPathways = c("[M+]","[M+H]","[M+H20+H]","[M+Na]")
Formula_adducts <- formula_adduct_calculator(molecular_formula, IonPathways)
```

formula_vector_generator *Molecular Formula Vector Generator*

Description

This function convert a molecular formulas into a numerical vector

Usage

```
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 <- "[13]C2C12H2Br5Cl3O"
Elements_molecular_formula <- c("[13]C", "C", "H", "O", "Br", "Cl")
EL <- element_sorter(ElementList = Elements_molecular_formula, alphabeticalOrder = TRUE)
Elements <- EL[["Elements"]]
LElements <- length(Elements)
##
mol_vec <- formula_vector_generator(molecular_formula, Elements, LElements,
allowedRedundantElements = TRUE)
##
regenerated_molecular_formula <- hill_molecular_formula_printer(Elements, mol_vec)
```

---

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

```
hill_molecular_formula_printer(Elements, MolVecMat, number_processing_threads = 1)
```

Arguments

Elements	A vector string of the used elements.
MolVecMat	A matrix of numerical vectors of molecular formulas in each row.
number_processing_threads	Number of processing threads for multi-threaded processing

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 <- hill_molecular_formula_printer(Elements, MolVecMat)
```

---

identificationScoreCalculator

Multiplicative Identification Score for the IDSL.UFA pipeline

---

Description

This function calculates the score values to rank candidate molecular formulas for a mass spectrometry-chromatography peak.

Usage

```
identificationScoreCalculator(scoreCoefficients, nIisotopologues, PCS, RCS, NEME,
R13C_PL, R13C_IP)
```

Arguments

`scoreCoefficients`	A vector of seven numbers equal or greater than 0
`nIisotopologues`	Number of isotopologues in the theoretical isotopic profiles.
`PCS`	PCS (per mille)
`RCS`	RCS (percentage)
`NEME`	NEME (mDa)
`R13C_PL`	R13C of the peak from IDSL.IPA peaklists
`R13C_IP`	R13C from theoretical isotopic profiles

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

```
ionization_pathway_deconvoluter(IonPathways, 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

Value

A list of adduct calculation values for each ionization pathway.

Examples

```
Elements <- element_sorter(alphabeticalOrder = TRUE)[["Elements"]]
IonPathways <- c("[M]+", "[M+H]+", "[2M-Cl]-", "[3M+CO2-H2O+Na-KO2+HCl-NH4]-")
Ion_DC <- ionization_pathway_deconvoluter(IonPathways, Elements)
```

isotopic_profile_calculator
Isotopic Profile Calculator

Description

This function was designed to calculate isotopic profile distributions for small molecules with masses <= 1200 Da. Nonetheless, this function may suit more complicated tasks with complex biological compounds. Details of the equations used in this function are available in the reference[1]. In this function, neighboring isotopologues are merged using the satellite clustering merging (SCM) method described in the reference[2].

Usage

```
isotopic_profile_calculator(MoleFormVec, massAbundanceList, peak_spacing,
intensity_cutoff, UFA_IP_memeory_variables = c(1e30, 1e-12, 100))
```

Arguments

`MoleFormVec`	A numerical vector of the molecular formula
`massAbundanceList`	A list of isotopic mass and abundance of elements obtained from the 'element_sorter' function
`peak_spacing`	A maximum space between two isotopologues in Da
`intensity_cutoff`	A minimum intensity threshold for isotopic profiles in percentage
`UFA_IP_memeory_variables`	A vector of three variables. Default values are c(1e30, 1e-12, 100) to manage memory usage. UFA_IP_memeory_variables[1] is used to control the overall size of isotopic combinations. UFA_IP_memeory_variables[2] indicates the minimum relative abundance (RA calculated by eq(1) in the reference [1]) of an isotopologue to include in the isotopic profile calculations. UFA_IP_memeory_variables[3] is the maximum elapsed time to calculate the isotopic profile on the 'setTimeLimit' function of base R.

Value

A matrix of isotopic profile. The first and second column represents the mass and intensity profiles, respectively.

References

[1] Fakouri Baygi, S., Crimmins, B.S., Hopke, P.K. Holsen, T.M. (2016). Comprehensive emerging chemical discovery: novel polyfluorinated compounds in Lake Michigan trout. *Environmental Science and Technology*, 50(17), 9460-9468, doi:10.1021/acs.est.6b01349.

[2] Fakouri Baygi, S., Fernando, S., Hopke, P.K., Holsen, T.M. and Crimmins, B.S. (2019). Automated Isotopic Profile Deconvolution for High Resolution Mass Spectrometric Data (APGC-QToF) from Biological Matrices. *Analytical chemistry*, 91(24), 15509-15517, doi:10.1021/acs.analchem.9b03335.

See Also

https://ipc.idsl.me/

Examples

```
EL <- element_sorter(alphabeticalOrder = TRUE)
Elements <- EL[["Elements"]]
massAbundanceList <- EL[["massAbundanceList"]]
peak_spacing <- 0.005 # mDa
intensity_cutoff <- 1 # (in percentage)
MoleFormVec <- formula_vector_generator("C8H10N4O2", Elements)
IP <- isotopic_profile_calculator(MoleFormVec, massAbundanceList, peak_spacing,
intensity_cutoff)
```

molecularFormula2IPdb *Molecular Formula to IPDB*

Description

A function to calculate IPDBs from a vector of molecular formulas

Usage

```
molecularFormula2IPdb(molecularFormulaDatabase, retentionTime = NULL, peak_spacing = 0,
intensity_cutoff_str = 1, IonPathways = "[M]+", number_processing_threads = 1,
UFA_IP_memeory_variables = c(1e30, 1e-12, 100), allowedMustRunCalculation = FALSE,
allowedVerbose = TRUE)
```

Arguments

molecularFormulaDatabase	A vector string of molecular formulas OR a list of elements and molecular formula matrix
retentionTime	retention time
peak_spacing	A maximum space between isotopologues in Da to merge neighboring isotopologues.
intensity_cutoff_str	A minimum intensity threshold for isotopic profiles in percentage. This parameter may be a string piece of R commands using c, b, br, cl, k, s, se, and si variables corresponding to the same elements.
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]-")'
number_processing_threads	number of processing cores for multi-threaded computations.
UFA_IP_memeory_variables	A vector of three variables. Default values are c(1e30, 1e-12, 100) to manage memory usage. UFA_IP_memeory_variables[1] is used to control the overall size of isotopic combinations. UFA_IP_memeory_variables[2] indicates the minimum relative abundance (RA calculated by eq(1) in the reference [1]) of an isotopologue to include in the isotopic profile calculations. UFA_IP_memeory_variables[3] is the maximum elapsed time to calculate the isotopic profile on the 'setTimeLimit' function of the base R.
allowedMustRunCalculation	c(TRUE, FALSE). A 'TRUE' allowedMustRunCalculation applies a brute-force method to calculate complex isotopic profiles. When 'TRUE', this option may significantly reduce the speed for multithreaded processing.
allowedVerbose	c(TRUE, FALSE). A 'TRUE' allowedVerbose provides messages about the flow of the function.

Value

An IPDB list of isotopic profiles

References

[1] Fakouri Baygi, S., Crimmins, B.S., Hopke, P.K. Holsen, T.M. (2016). Comprehensive emerging chemical discovery: novel polyfluorinated compounds in Lake Michigan trout. *Environmental Science and Technology*, 50(17), 9460-9468, doi:10.1021/acs.est.6b01349.

See Also

https://ipc.idsl.me/

Examples

```
library(IDSL.UFA, attach.required = TRUE)
molecular_formula <- c("C13F8N8O2", "C20H22", "C8HF16ClSO3", "C12Cl10", "C123H193N35O37")
peak_spacing <- 0.005 # in Da for QToF instruments
# Use this piece of code for intensity cutoff to preserve significant isotopologues
intensity_cutoff_str <- "if (s>0 & si>0) {min(c(c, 10, si*3, s*4))}
else if (s>0 & si==0) {min(c(c, 10, s*4))}
else if (s==0 & si>0) {min(c(c, 10, si*3))}
else if (s==0 & si==0) {min(c(c, 10))}"
UFA_IP_memeory_variables <- c(1e30, 1e-12, 100)
IonPathways <- c("[M+H]+", "[M+Na]+", "[M-H2O+H]+")
number_processing_threads <- 2
listIsoProDataBase <- molecularFormula2IPdb(molecular_formula, retentionTime = NULL,
peak_spacing, intensity_cutoff_str, IonPathways, number_processing_threads,
UFA_IP_memeory_variables, allowedMustRunCalculation = FALSE, allowedVerbose = TRUE)
save(listIsoProDataBase, file = "listIsoProDataBase.Rdata")
```

---

molecular_formula_annotator

Molecular Formula Annotator

Description

This module annotates candidate molecular formulas in the peaklists from the IDSL.IPA pipeline using isotopic profiles.

Usage

```
molecular_formula_annotator(IPDB, spectraList, peaklist, selectedIPApeaks,
massAccuracy, maxNEME, minPCS, minNDCS, minRCS, scoreCoefficients, RTtolerance = NA,
correctedRTpeaklist = NULL, exportSpectraParameters = NULL, number_processing_threads = 1)
```

Arguments

IPDB	An isotopic profile database produced by the IDSL.UFA functions.
spectraList	a list of mass spectra in each chromatogram scan.
peaklist	Peaklist from the IDSL.IPA pipeline
selectedIPApeaks	selected IPA peaklist
massAccuracy	Mass accuracy in Da
maxNEME	Maximum value for Normalized Euclidean Mass Error (NEME) in mDa
minPCS	Minimum value for Profile Cosine Similarity (PCS)
minNDCS	Minimum value for Number of Detected Chromatogram Scans (NDCS)
minRCS	Minimum value for Ratio of Chromatogram Scans (RCS) in percentage
scoreCoefficients	A vector of five numbers representing coefficients of the identification score
RTtolerance	Retention time tolerance (min)
correctedRTpeaklist	corrected retention time peaklist
exportSpectraParameters	Parameters for export MS/MS match figures
number_processing_threads	Number of processing threads for multi-threaded processing

Value

A dataframe of candidate molecular formulas

---

molecular_formula_elements_filter

molecular_formula_elements_filter

Description

molecular_formula_elements_filter

Usage

```
molecular_formula_elements_filter(molecularFormulaMatrix, Elements)
```

Arguments

molecularFormulaMatrix	molecularFormulaMatrix
Elements	Elements

Value

a list of molecularFormulaMatrix and elementSorterList.

molecular_formula_library_generator
Molecular Formula Database Producer

Description

This function generates an efficient database for molecular formula matching against a database.

Usage

```
molecular_formula_library_generator(entire_molecular_formulas)
```

Arguments

entire_molecular_formulas
A string vector of molecular formulas (redundancy is allowed)

Value

A vector of frequency of molecular formulas in the database.

Examples

```
entire_molecular_formulas <- c("C2H6O", "C2H6O", "C2H6O", "C2H6O", "CH4O", "CH4O",
"CH4O", "NH4", "C6H12O6")
db <- molecular_formula_library_generator(entire_molecular_formulas)
freq <- db[c("C6H12O6", "CH4O")]
```

molecular_formula_library_search
Molecular Formula Library Search

Description

This function attempts to match candidate molecular formulas against a library of molecular formulas using a set of ionization pathways.

Usage

```
molecular_formula_library_search(MolecularFormulaAnnotationTable, MFlibrary,
IonPathways, number_processing_threads = 1)
```

Arguments

MolecularFormulaAnnotationTable	A molecular formula annotation table from the 'molecular_formula_annotator' module.
MFlibrary	A library of molecular formulas generated using the 'molecular_formula_library_generator' module.
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]-")'
number_processing_threads	Number of processing threads for multi-threaded processing

---

monoisotopicMassCalculator

Monoisotopic Mass Calculator

Description

This function calculates monoisotopic mass of a molecular formula

Usage

```
monoisotopicMassCalculator(MoleFormVec, massAbundanceList,
LElements = length(massAbundanceList))
```

Arguments

MoleFormVec	A numerical vector molecular formula
massAbundanceList	A list of isotopic mass and abundance of elements obtained from the 'element_sorter' function
LElements	length of elements

Value

The monoisotopic mass

Examples

```
Elements <- c("C", "H", "O")
MoleFormVec <- c(2, 6, 1) # C2H6O
EL_mass_abundance <- element_sorter(ElementList = Elements, alphabeticalOrder = FALSE)
massAbundanceList <- EL_mass_abundance[["massAbundanceList"]]
MImass <- monoisotopicMassCalculator(MoleFormVec, massAbundanceList)
```

scoreCoefficientsEvaluation
Score Coefficient Evaluation

Description

This function evaluates the efficiency of the optimization process.

Usage

```
scoreCoefficientsEvaluation(PARAM_ScoreFunc)
```

Arguments

PARAM_ScoreFunc
PARAM_ScoreFunc is a variable derived from the 'UFA_coefficient_xlsxAnalyzer' function

scoreCoefficientsOptimization
Coefficients Score Optimization

Description

This function optimizes the coefficients of the score function.

Usage

```
scoreCoefficientsOptimization(PARAM_ScoreFunc)
```

Arguments

PARAM_ScoreFunc
PARAM_ScoreFunc is a variable derived from the 'UFA_score_function_optimization_xlsxAnalyzer' function

scoreCoefficientsReplicate *Zero Score Function*

Description

This function generates the input for the score optimization.

Usage

```
scoreCoefficientsReplicate(PARAM_ScoreFunc)
```

Arguments

PARAM_ScoreFunc
: PARAM_ScoreFunc is a variable derived from the 'UFA_coefficient_xlsxAnalyzer' function

UFA_enumerated_chemical_space *IPDBs from UFA Enumerated Chemical Space (ECS) approach*

Description

This function produces the isotopic profile database using the UFA enumerated chemical space (ECS) approach.

Usage

```
UFA_enumerated_chemical_space(PARAM_ECS)
```

Arguments

PARAM_ECS	A dataframe of the molecular formula constraints in the UFA spreadsheet

UFA_enumerated_chemical_space_xlsxAnalyzer *IPDBs from UFA Enumerated Chemical Space (ECS) xlsx Analyzer*

Description

This function evaluates the molecular formula generation constraints in the spreadsheet to create the isotopic profile database.

Usage

```
UFA_enumerated_chemical_space_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	UFA spreadsheet

UFA_formula_source *IPDB from a Molecular Formulas Source*

Description

This function produces IPDB from a molecular formula source (a .csv/.txt/.xlsx file).

Usage

```
UFA_formula_source(PARAM_FormSource)
```

Arguments

PARAM_FormSource	PARAM_FormSource is an internal variable of the IDSL.UFA package.

Value

an IPDB is saved in the destination address

UFA_formula_source_xlsxAnalyzer *UFA Formula Source xlsxAnalyzer*

Description

This function evaluates the parameter spreadsheet for score coefficients optimization.

Usage

```
UFA_formula_source_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	The parameter spreadsheet in the .xlsx format.

Value

a processed parameter to feed the 'UFA_molecular_formulas_source' function.

UFA_IPdbMerger *UFA IPDB Merger*

Description

To merge multiple IPDBs into one IPDB

Usage

```
UFA_IPdbMerger(path, vecIPDB)
```

Arguments

path	path
vecIPDB	a vector of IPDBs

Value

IPDB

UFA_locate_regex *UFA Locate regex*

Description

Locate indices of the pattern in the string

Usage

```
UFA_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 <- UFA_locate_regex(string, pattern)
```

UFA_PubChem_formula_extraction

UFA PubChem Formula Extraction

Description

This module is to extract molecular formulas from a database.

Usage

```
UFA_PubChem_formula_extraction(path)
```

Arguments

path	path to store information

Value

Molecular formulas in https://pubchem.ncbi.nlm.nih.gov/ are stored in the provided 'path' in the .txt format.

References

Kim S, Chen J, Cheng T, Gindulyte A, He J, He S, Li Q, Shoemaker BA, Thiessen PA, Yu B, Zaslavsky L, Zhang J, Bolton EE. PubChem in 2021: new data content and improved web interfaces. *Nucleic Acids Res.*, 2021 Jan 8; 49(D1):D1388–D1395 , doi:10.1093/nar/gkaa971.

UFA_score_coefficients_corrector

Score Coefficients Corrector for MolecularFormulaAnnotationTable

Description

This function updates ranking orders of the individual MolecularFormulaAnnotationTable when score coefficients changed.

Usage

```
UFA_score_coefficients_corrector(input_annotated_molf_address,
output_annotated_molf_address, scoreCoefficients, number_processing_threads = 1)
```

Arguments

`input_annotated_molf_address`
: Address to load the individual MolecularFormulaAnnotationTables.

`output_annotated_molf_address`
: Address to save the individual MolecularFormulaAnnotationTables.

`scoreCoefficients`
: A vector of five numbers representing coefficients of the identification score function.

`number_processing_threads`
: Number of processing threads for multi-threaded computations.

`UFA_score_function_optimization` *UFA Score Coefficient Workflow*

Description

This function runs the score optimization workflow.

Usage

```
UFA_score_function_optimization(PARAM_ScoreFunc)
```

Arguments

`PARAM_ScoreFunc`
: PARAM_ScoreFunc from the 'UFA_score_function_optimization_xlsxAnalyzer' module

---

`UFA_score_function_optimization_xlsxAnalyzer` *UFA Score Optimization xlsx Analyzer*

Description

This function evaluates the parameter spreadsheet for score coefficients optimization.

Usage

```
UFA_score_function_optimization_xlsxAnalyzer(spreadsheet)
```

Arguments

`spreadsheet` The parameter spreadsheet in the .xlsx format.

Value

a processed parameter to feed the 'UFA_score_function_optimization' function.

UFA_workflow *UFA Workflow*

Description

This function executes the UFA workflow in order.

Usage

```
UFA_workflow(spreadsheet)
```

Arguments

spreadsheet	UFA spreadsheet

Value

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

UFA_xlsxAnalyzer *UFA xlsx Analyzer*

Description

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

Usage

```
UFA_xlsxAnalyzer(spreadsheet)
```

Arguments

spreadsheet	UFA spreadsheet

Value

This function returns the UFA parameters to feed the UFA_workflow function.

Index