Package: RuSirius (via r-universe)

June 3, 2026

Title R Implementation of the Sirius software

Version 1.0.2

Description The RuSirius package allows the user to interact with the Sirius software from R. Sirius is a software for the analysis and annotation of mass spectrometry data. RuSirius makes use of the API present in RSirius. It is dependent on Sirius 6.4, please ensure you have the right version downloaded.

Depends ProtGenerics (>= 1.35.4), R (>= 4.4.0), RSirius, Spectra (>= 1.17.3)

Imports jsonlite, methods, MetaboAnnotation, dplyr

Suggests testthat, knitr (>= 1.1.0), rmarkdown, BiocStyle, xcms (>= 4.5.2), MsExperiment, MsDataHub

License Artistic-2.0

Encoding UTF-8

biocViews MassSpectrometry, Metabolomics, Annotation, Software

Roxygen list(markdown=TRUE)

BugReports https://github.com/RforMassSpectrometry/RuSirius/issues

URL https://github.com/RforMassSpectrometry/RuSirius

Remotes sirius-ms/sirius-client-openAPI/client-api_r/generated@6470a73

VignetteBuilder knitr

Collate 'Sirius.R' 'Utils.R' 'helpers.R' 'import.R' 'results.R' 'run-param.R' 'run.R' 'siriusDbs.R'

Config/roxygen2/version 8.0.0

Repository https://rformassspectrometry.r-universe.dev

Date/Publication 2026-05-21 15:47:22 UTC

RemoteUrl https://github.com/rformassspectrometry/RuSirius

RemoteRef HEAD

RemoteSha 936cb875b37ea7861aad1a64204448ca65d24207

Contents

`deNovoStructureParam` *de novo structure annotation*

Description

This function to set up the parameter for de novo structure annotation using the *MSNovelist* tool.

MSNovelist generates molecular structures de novo from MS/MS data - without relying on any database. This makes it particularly useful for analyzing poorly represented analyte classes and novel compounds, where traditional database searches may fall short. However, it is not intended to replace database searches altogether, as structural elucidation of small molecules from MS/MS data remains a challenging task, and identifying a structure without database candidates is even more difficult.

Usage

```
deNovoStructureParam(numberOfCandidateToPredict = 10)
```

Arguments

`numberOfCandidateToPredict`
: `numeric`, number of structure candidates to be predicted by MsNovelist. Max Value `128`. Actual number of returned candidate might be lower du to duplicates being created. Default is `10`

Value

An object of class `deNovoStructureParam`.

Note

For more information, see the Sirius documentation.

References

Stravs MA, Dührkop K, Böcker S, Zamboni N (2022). MSNovelist: de novo structure generation from mass spectra. *Nature Methods*, 19, 865-870. doi:10.1038/s41592022014863

Examples

```
# Example of setting up the parameters for de novo structure annotation
param <- deNovoStructureParam(numberOfCandidateToPredict = 10)
```

`formulaIdParam` *Identifying molecular formula*

Description

This function configures the parameters for molecular formula annotation in Sirius. Molecular formula identification is done using isotope pattern analysis on the MS1 data as well as fragmentation tree computation on the MS2 data. The score of a molecular formula candidate is a combination of the isotope pattern score and the fragmentation tree score.

Usage

```
formulaIdParam(
  instrument = c("QTOF", "ORBITRAP", "FTICR"),
  numberOfCandidates = 10,
  numberOfCandidatesPerIonization = 1,
  massAccuracyMS2ppm = 10,
  isotopeMs2Settings = c("IGNORE", "FILTER", "SCORE"),
  filterByIsotopePattern = TRUE,
  enforceElGordoFormula = TRUE,
  performBottomUpSearch = TRUE,
  performDeNovoBelowMz = 400,
  formulaSearchDBs = character(0),
  applyFormulaConstraintsToDBAndBottomUpSearch = FALSE,
  enforcedFormulaConstraints = c("H", "C", "N", "O", "P"),
  fallbackFormulaConstraints = c("S"),
  detectableElements = c("B", "S", "Cl", "Se", "Br"),
  ilpTimeout = FALSE,
  numberOfSecondsPerDecomposition = 0,
  numberOfSecondsPerInstance = 0,
  useHeuristic = TRUE,
  useHeuristicAboveMz = 300,
  useOnlyHeuristicAboveMz = 650,
  injectSpecLibMatchFormulas = TRUE,
  minScoreToInjectSpecLibMatch = 0.7,
  minPeaksToInjectSpecLibMatch = 6,
  candidateFormulas = character(0)
)
```

Arguments

`instrument` `character(1)` The type of mass spectrometer used for the analysis. Options include `"QTOF"`, `"ORBITRAP"`, and `"FTICR"`. This choice mainly affects the allowed mass deviation. If you are unsure about the instrument, use the default value `"QTOF"`.

`numberOfCandidates`
`integer(1)` The number of formula candidates to keep in the result list. Default is `10`.

`numberOfCandidatesPerIonization`
`integer(1)` Forces SIRIUS to report at least this number of candidates per ionization.

`massAccuracyMS2ppm`
`numeric(1)` The maximum allowed mass deviation (in parts per million, ppm) for molecular formulas. Only formulas within this mass window are considered. Default is `10`.

`isotopeMs2Settings`
`character(1)` Specifies how isotope patterns in MS/MS should be handled. Default is `"IGNORE"`. Options:

- `` `"FILTER"` ``: Excludes formulas if the theoretical isotope pattern doesn't match.
- `` `"SCORE"` ``: Uses isotope patterns for scoring, useful for clear MS/MS isotope patterns.
- `` `"IGNORE"` ``: Ignores isotope patterns in MS/MS.

`filterByIsotopePattern`
`logical` When `TRUE`, filters molecular formulas by comparing their theoretical isotope patterns to the measured ones, excluding those that don't match. Default is `TRUE`.

`enforceElGordoFormula`
`logical` El Gordo may predict that an MS/MS spectrum is a lipid spectrum. If enabled, the corresponding molecular formula will be enforeced as molecular formula candidate. Default is `TRUE`.

`performBottomUpSearch`
`logical` If `TRUE`, enables molecular formula generation through a bottom-up search. Default is `TRUE`.

`performDeNovoBelowMz`
`numeric(1)` Specifies the m/z below which de novo molecular formula generation is enabled. Set to `0` to disable de novo molecular formula generation. Default is `400`.

`formulaSearchDBs`
`list(character)` A list of structure databases (e.g., `"CHEBI"`, `"HMDB"`) from which molecular formulas are extracted to reduce the search space. Use only if necessary, as de novo formula annotation is usually more effective. Default is `character(0)`.

`applyFormulaConstraintsToDBAndBottomUpSearch`
`logical` If `TRUE`, applies formula (element) constraints to both database search and bottom-up search, in addition to de novo generation. Default is `FALSE`.

`enforcedFormulaConstraints`
: `character` Specifies the elements that are always considered when auto-detecting the formula. Enforced elements are always included in the formula, even if the compound is already assigned to a specific molecular formula. Default is `H,C,N,O,P`.

`fallbackFormulaConstraints`
: `character` Specifies the elements that are used as fallback when auto-detection fails (e.g., no isotope pattern). Default is `S`.

`detectableElements`
: `list(character)` Defines the elements that can be added to the chemical alphabet when detected in the spectrum, such as from isotope patterns. Default is `c("B", "S", "Cl", "Se", "Br")`.

`ilpTimeout`
: `logical` The timeout settings for the integer linear programming (ILP) solver. If `TRUE`, it should include timeout parameters such as `numberOfSecondsPerDecomposition` and `numberOfSecondsPerInstance`.

`numberOfSecondsPerDecomposition`
: `numeric`

`numberOfSecondsPerInstance`
: `numeric`

`useHeuristic`
: `logical` If `TRUE`, enables the use of heuristics in molecular formula annotation. When enabled, additional thresholds like `useHeuristicAboveMz` and `useOnlyHeuristicAboveMz` can be set.

`useHeuristicAboveMz`
: `numeric` The m/z threshold above which heuristic is used. Default is `300`.

`useOnlyHeuristicAboveMz`
: `numeric(1)` The m/z threshold above which only heuristic is used. Default is `650`.

`injectSpecLibMatchFormulas`
: `logical` If `TRUE`, formula candidates matching spectral library entries above a certain similarity threshold will be preserved for further analysis, regardless of score or filter settings. Default is `TRUE`.

`minScoreToInjectSpecLibMatch`
: `numeric(1)` The similarity threshold for injecting spectral library match formulas. If the score is above this threshold, the formula will be preserved. Default is `0.7`.

`minPeaksToInjectSpecLibMatch`
: `integer` The minimum number of matching peaks required to inject spectral library match formulas into further analysis.

`candidateFormulas`
: `character` Optional vector of molecular formulas to use as candidates. When provided, SIRIUS will only consider these formulas instead of performing de novo formula generation. This is useful when the molecular formula of a compound is already known. Formulas should be given as neutral molecular formulas (e.g., `"C10H12N2O"`, `"C6H12O6"`). You can provide one or more formulas. Default is `character(0)` (no restriction, all formulas are considered).

Value

An object of class `formulaIdParam` with the specified parameters.

General parameters

We advise to set up these following parameter to fit your specific study.

Advanced parameters

If you want to specify these parameters we advise you read the Sirius documentation to learn how to adapt them to your dataset and annotation needs.

Note

For more information, see the Sirius documentation.

References

Dührkop K, Fleischauer M, Ludwig M, et al. (2019). SIRIUS 4: a rapid tool for turning tandem mass spectra into metabolite structure information. *Nature Methods*, 16, 299-302. doi:10.1038/s4159201903448

Examples

```
# Example of creating a formulaIdParam object
param <- formulaIdParam(instrument = "QTOF",
                        numberOfCandidates = 5,
                        enforceElGordoFormula = TRUE)

# Restrict formula identification to a known molecular formula
param_known <- formulaIdParam(candidateFormulas = "C10H12N2O")
```

---

import *Import Data into Sirius*

Description

Import Data into Sirius

Usage

```
import(
  sirius,
  spectra,
  ms_column_name = character(),
  adducts = character(),
  chunkSize = 500,
  deleteExistingFeatures = TRUE
)
```

Arguments

sirius	Sirius. The connection to the Sirius instance with a loaded project.
spectra	Spectra object containing MS data to import. Can contain MS1 and/or MS2 spectra. If multiple MS levels are present, the ms_column_name parameter must be provided to group spectra into features. See the fragmentGroupIndex() function from the Spectra package for generating appropriate grouping indices.
ms_column_name	character(1). This is the name of a column expected in the spectraData of the input spectra object. MS1 and MS2 spectra with the same index in this column will be grouped into a single feature. This parameter must be provided if the object has multiple msLevels. If no column name is provided and the object contains only one MS level, then each MS1 spectrum will be exported as a separate feature in Sirius. Lastly, we support multiple MS2 spectra per feature, but only one MS1 spectrum per index. Please merge MS1 spectra beforehand if necessary.
adducts	character vector of the adduct(s) associated with the features being imported. Must be either length 1 or the same length as the number of features. If of length 1 and fewer than the number of features, the same adduct will be used for all features.
chunkSize	numeric(1). Number of features to process and import at once. Importing can be slow, so this is useful when working with a very large number of spectra.
deleteExistingFeatures	logical(1). If TRUE, all existing features will be deleted before importing the new ones.

Value

A Sirius object with the imported features and updated feature map.

---

predictParam *Predicting FingerPrint and compounds Identifications*

Description

This function creates an object of class predictParam that can be used to predict molecular fingerprints and compound identifications using *CSI:FingerID* and *CANOPUS*.

CSI:FIngerID identifies the structure of a molecule by predicting its molecular fingerprint and using this fingerprint to search in a molecular structure database.

CANOPUS (Dührkop et al.) predicts the presense/absense of more than 2500 compound classes. CANOPUS predicts these classes based solely on MS/MS data and without requiring database information. This means it can identify a class even if no molecular structure of that class exists in the molecular structure database.

Usage

```
predictParam(useScoreThreshold = TRUE, alwaysPredictHighRefMatches = FALSE)
```

Arguments

useScoreThreshold
: logical whether to use a soft threshold to be applied to only compute FingerPrints for promising formula candidates. Enabling is highly recommended. Default is TRUE.

alwaysPredictHighRefMatches
: logical wether to predict FingerPrint/Classes&Structure for formulas candidates with reference spectrum similarity > Sirius.minReferenceMatchScoreToInject no matter which score threshold rules apply. Default is FALSE.

Value

An object of class predictParam with the specified parameters.

Note

For more information, see the Sirius documentation.

References

Dührkop K, Shen H, Meusel M, et al. (2015). Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proceedings of the National Academy of Sciences*, 112, 12580-12585. doi:10.1073/pnas.1509788112

Dührkop K, Nothias L-F, Fleischauer M, et al. (2021). Systematic classification of unknown metabolites using high-resolution fragmentation mass spectra. *Nature Biotechnology*, 39, 462-471. doi:10.1038/s4158702007408

Examples

```
# Example of setting up the parameters for the prediction of molecular
# fingerprints and compound class
param <- predictParam()
```

---

results *Fetch Results from Sirius*

Description

Functions to retrieve results from the *Sirius* project, allowing customization of the result type, feature selection, and return format.

Get best match results for a Sirius project. Choose which type of results to fetch with the *result.type* parameter. "structure" and "deNovo" will both also give information on the formula.

Usage

```
## S4 method for signature 'Sirius'
summary(
  object,
  result.type = c("formulaId", "structure", "deNovo", "spectralDbMatch"),
  ...
)

results(
  sirius,
  features = character(),
  result.type = c("formulaId", "structureDb", "compoundClass", "deNovo",
    "spectralDbMatch", "fragTree"),
  return.type = c("list", "data.frame"),
  topFormula = 5,
  topStructure = 5,
  topSpectralMatches = 5
)
```

Arguments

`object`	A `Sirius` object.
`result.type`	`character(1)` specifying the type of results to fetch for the summary. Options are "formulaId", "structure", "deNovo", and "spectralDbMatch". Defaults to "formulaId".
`...`	Additional arguments (currently ignored).
`sirius`	A `Sirius` object.
`features`	`character()` vector specifying feature IDs to retrieve results for. Defaults to all available features.
`return.type`	`character(1)` specifying return format. Either "data.frame" (default) or "list".
`topFormula`	`numeric(1)` Maximum number of formula candidates to retrieve per feature. Defaults to 5.
`topStructure`	`numeric(1)` Maximum number of structure candidates per formula. Defaults to 5.
`topSpectralMatches`	`numeric(1)` Maximum number of spectral matches per feature. Defaults to 5.

Value

A data.frame or list of results, depending on `return.type`.

A `data.frame` with the summary of the results. Important column is the *ApproximateConfidence* column, which gives a score of how possible all the identifications for this feature are. The *exactConfidence* column is a score of how possible the top identification is.

run *Run job on Sirius.*

Description

This function configures the job submission to Sirius. It creates an object of class `config` that can be used to submit a job Sirius, it can also be saved and reused later on through the `saveJobConfig()`function. For example on how to use, see the vignette.

Depending on what task you want to perform, you can specify the following parameters:

- `spectraMatchingParam`: Allows to perform matching betweeen spectra input to spectral libraries.
- `formulaIdParam`: Allows to generate molecular formula candidates for each features.
- `zodiacParam`: Allows to perform re-ranking of formula candidates using *Zodiac*. It is advised to only perform it if De Novo structure annotation is run later.
- `predictParam`: Allows to perform molecular fingeprint prediction using *CSI:FingerID* and compound classification using *CANOPUS*.
- `structureDbSearchParam`: Allows to perform structure annotation based on the fingerprint identifications.
- `deNovoStructureParam`: Allows to perform de novo structure generation using the *MSNovelist* tool.

Usage

```
run(
  sirius,
  compoundsIds = character(),
  alignedFeaturesIds = featuresId(sirius),
  fallbackAdducts = c("[M + H]+", "[M - H]-", "[M + Na]+", "[M + K]+"),
  enforceAdducts = character(),
  detectableAdducts = c("[M + H3N + H]+", "[M - H4O2 + H]+", "[M - H2O - H]-",
    "[M - H3N - H]-", "[M + Cl]-", "[2M + K]+", "[M + K]+", "[2M + Cl]-",
    "[M + C2H4O2 - H]-", "[M + H]+", "[2M + H]+", "[M - CH3 - H]-", "[M - H]-",
    "[M + Na]+", "[M - H2O + H]+"),
  spectraSearchParams = NA,
  formulaIdParams = NA,
  zodiacParams = NA,
  predictParams = NA,
  structureDbSearchParams = NA,
  msNovelistParams = NA,
  recompute = FALSE,
  configFile = character(),
  wait = TRUE
)
```

```
config(
  compoundsIds = character(),
  alignedFeaturesIds = character(),
  fallbackAdducts = c("[M + H]+", "[M - H]-", "[M + Na]+", "[M + K]+"),
  enforceAdducts = character(),
  detectableAdducts = c("[M + H3N + H]+", "[M - H4O2 + H]+", "[M - H2O - H]-",
    "[M - H3N - H]-", "[M + Cl]-", "[2M + K]+", "[M + K]+", "[2M + Cl]-",
    "[M + C2H4O2 - H]-", "[M + H]+", "[2M + H]+", "[M - CH3 - H]-", "[M - H]-",
    "[M + Na]+", "[M - H2O + H]+"),
  formulaIdParams = formulaIdParam(),
  zodiacParams = NA,
  predictParams = NA,
  structureDbSearchParams = NA,
  msNovelistParams = NA,
  spectraSearchParams = NA,
  recompute = FALSE
)
```

Arguments

`sirius`	`Sirius` object, the connection to the Sirius server.
`compoundsIds`	`character` vector, the ids of the compounds to process.
`alignedFeaturesIds`	`character` vector, the ids of the aligned features to process. By default, computes all.
`fallbackAdducts`	`character` vector, fallback adducts are considered if the auto detection did not find any indication for an ion mode.
`enforceAdducts`	`character` vector, the adducts to enforce. They are always considered.
`detectableAdducts`	`character` vector, detectable adducts are only considered if there is an indication in the MS1 scan (e.g. correct mass delta).
`spectraSearchParams`	object of class `spectraMatchingParam`, containing the parameters for the spectra matching.
`formulaIdParams`	object of class `formulaIdParam`, containing the parameters for the molecular formula identification.
`zodiacParams`	object of class `zodiacParam`, containing the parameters for the Zodiac re-ranking.
`predictParams`	object of class `predictParam`, containing the parameters for the molecular fingerprint prediction and compound classification.
`structureDbSearchParams`	object of class `structureDbParam`, containing the parameters for the structure annotation.
`msNovelistParams`	object of class `deNovoStructureParam`, containing the parameters for the de novo structure generation.

recompute	logical, whether to recompute the job , only necessary if the configfile comes from a saved file. Default is FALSE.
configFile	character, the path to the configuration file to use for the job submission, optional.
wait	logical, whether to wait for the job to finish. Default is TRUE

Value

The job ID of the submitted job, it can be inputted in the jobInfo() function to retrieve the job information. To retrieve results see results documentation.

Only formula identification

If you only want to perform formula identification, you can by only inputing the formulaIdParam object. In combination, you can also input the spectraMatchingParam object to perform spectral matching and subsequently compare the results.

Known molecular formula

If you already know the molecular formula for a compound, you can restrict SIRIUS to only consider that formula using the candidateFormulas parameter of formulaIdParam. This skips the de novo formula generation and instead computes the fragmentation tree for the specified formula(s) directly. Use the enforceAdducts parameter of run() to also fix the adduct type.

Structure annotation

To performe structure annotation, you need input the formulaIdParam object, as well as the predictParam object to perform molecular fingerprint prediction and compound classification. These results will then subsequently be used to perform structure annotation using the structureDbSearchParam object.

De Novo structure annotation

To perform de novo structure annotation, you need to input the formulaIdParam object, it is also advised in this case to perform re-ranking using the zodiacParam object. The molecular fingerprint prediction and compound classification can be performed using the predictParam object. The deNovoStructureParam object is then used to perform the de novo structure annotation.

Sirius *Connection to a Sirius instance*

Description

Creates a Sirius instance and checks that connection to the server is valid. returns the Api and SDK within its slots. Main object that the user will interact with to connect to the Sirius server and perform operations.

Creates a Sirius object and checks that the connection to the Sirius server is valid. If the Sirius server is not running, the function will attempt to start it using the provided path to the executable. If the connection is not valid, the function will attempt to log in using the provided credentials. If the connection is still not valid, the function will stop with an error message.

Usage

```
Sirius(
  username = character(),
  password = character(),
  projectId = character(),
  path = character(),
  port = integer(),
  path_to_sirius = character(),
  verbose = FALSE
)

## S4 method for signature 'Sirius'
show(object)
```

Arguments

username	character(1), the username to use for the connection
password	character(1), the password to use for the connection
projectId	character(1), the project id to use for the connection
path	character(1) path where to find the existing project or where to create a new one. By default, the project will be opened in the current "." directory.
port	integer(1) defining the port for Sirius. Allows to manually define the port to connect to. If not specified (the default) the port information will be read from a Sirius-config file.
path_to_sirius	character(1), optional path to the Sirius executable. Default is character(), in which case the executable is expected to be available on the system PATH. Provide this when running on a cluster or any environment where the executable is in a non-standard location.
verbose	logical(1), if TRUE the function will print all messages to the console. Use if need debug, default is FALSE.
object	Sirius, the object to show,

Value

Sirius object with the Sirius api connected.

Slots

`api ANY`, the api object to use for the connection

`sdk ANY`, the sdk object to use for the connection

`projectId character`, the project id to use for the connection

`featureMap data.frame`, the feature map to use for the connection

Author(s)

Philippine Louail (+people that worked on the API)

`siriusDbs` *Functions for Handling Sirius Databases*

Description

This set of functions provides tools for managing databases in Sirius. These include listing available databases, retrieving database information, deleting databases, and creating new databases from files.

List the databases that are searchable by Sirius for spectral matching.

Retrieve detailed information about a specific database.

Delete a database from Sirius.

Create a new database in Sirius from specified files.

Usage

```
listDbs(sirius)

infoDb(sirius, databaseId = character())

removeDb(sirius, databaseId = character())

createDb(
  sirius,
  databaseId = character(),
  files = character(),
  location = getwd()
)
```

Arguments

`sirius`	A `Sirius` object representing the Sirius connection.
`databaseId`	A `character(1)` string specifying the new database ID.
`files`	A `character` vector of file paths to add to the database.
`location`	A `character(1)` string specifying the location for the database. Defaults to the current working directory.

Value

A `data.frame` containing details of searchable databases.

A `list` containing details of the specified database.

A `logical(1)` indicating whether the database was successfully removed.

A `list` containing details of the created database.

`spectraMatchingParam` *Spectra database matching*

Description

This function is to set up the parameter for matching to spectra databases. This needs to be run first. Spectral library matching is performed using the cosine score with squared peak intensities, ignoring the precursor peak.

Note that spectral library matches are added as annotations to CSI:FingerID results and do not influence the ranking of structure candidates.

Usage

```
spectraMatchingParam(
  spectraSearchDBs = c("BIO", "massbank"),
  peakDeviationPpm = 10,
  precursorDeviationPpm = 10,
  scoring = c("MODIFIED_COSINE", "INTENSITY", "GAUSSIAN")
)
```

Arguments

`spectraSearchDBs`
: `character` vector of the databases to search. The default is `c("BIO", "massbank")`. Other values can be found by running `listDatabases()`.

`peakDeviationPpm`
: `numeric` Maximum allowed mass deviation in ppm for matching peaks.

`precursorDeviationPpm`
: `numeric` Maximum allowed mass deviation in ppm for matching the precursor. If not specified, the same value as for the peaks is used.

`scoring`
: `character` The scoring function to use. Possible values are `MODIFIED_COSINE`, `GAUSSIAN`, `INTENSITY`. The default is `MODIFIED_COSINE`.

Value

An object of class `spectraMatchingParam`

Note

For more information, see the Sirius documentation.

Examples

```
# Example of setting up the parameters for spectra matching
param <- spectraMatchingParam(spectraSearchDBs = c("BIO", "massbank"),
                              peakDeviationPpm = 10,
                              precursorDeviationPpm = 10,
                              scoring = "MODIFIED_COSINE")
```

`structureDbSearchParam` *Structure Database Search*

Description

This function creates an object of class `structureDbSearchParam` that can be used to configure the structure database search in Sirius. The object can be passed to the `runSirius` function to perform the structure database search.

By default, *SIRIUS* searches for molecular structures in a biomolecule structure database. It can also search in the (extremely large) PubChem database or in custom "suspect databases" provided by the user.

Usage

```
structureDbSearchParam(
  structureSearchDbs = c("BIO", "massbank"),
  tagStructuresWithLipidClass = TRUE,
  expansiveSearchConfidenceMode = c("APPROXIMATE", "EXACT", "OFF")
)
```

Arguments

`structureSearchDbs`
: `character` vector, specifying the id of the databases to search for structures. Default is `c("BIO", "massbank")`. If expansive search is enabled this database selection will be expanded to PubChem if not high confidence hit was found in the selected databases. The available databases can be found by running `listDatabases()`.

`tagStructuresWithLipidClass`
: `logical`, specifying whether to tag structures with lipid class estimated by El Gordo. The lipid class will only be available if El Gordo predicts that the MS/MS is a lipid spectrum. If this parameter is set to `FALSE` El Gordo will still be executed and e.g. improve the fragmentation tree, but the matching structure candidates will not be tagged if they match lipid class.

`expansiveSearchConfidenceMode`
: `character(1)` , specifying the confidence mode for expansive search. Possible values are `"APPROXIMATE"`,`"EXACT"`, `"OFF"`. Defaults to `"APPROXIMATE"`.

Value

An object of class `structureDbSearchParam`.

Note

For more information, see the Sirius documentation.

References

Dührkop K, Shen H, Meusel M, et al. (2015). Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proceedings of the National Academy of Sciences*, 112, 12580-12585. doi:10.1073/pnas.1509788112

Examples

```
# Example of setting up the parameters for structure database search
param <- structureDbSearchParam(
                structureSearchDbs = c("BIO", "massbank"),
                tagStructuresWithLipidClass = TRUE,
                expansiveSearchConfidenceMode = "APPROXIMATE")
```

`utils` *Utility function for RuSirius*

Description

returns `TRUE` if the connection to the Sirius is valid, `FALSE` otherwise.

Usage

```
logIn(sirius, username, password)

checkConnection(sirius)

shutdown(sirius, closeProject = TRUE)

openGUI(sirius)

closeGUI(sirius, closeProject = FALSE)

projectInfo(sirius, infoType = c("compatibilityInfo", "sizeInformation"))

listOpenProjects(sirius)

openProject(sirius, projectId, path = character())
```

```
featuresId(sirius, type = c("sirius", "xcms"))

featuresInfo(sirius)

deleteFeatures(sirius, featureId = featuresId(sirius))

mapFeatures(sirius)

saveConfig(sirius, config, name)

jobInfo(sirius, jobId = character())

deleteJob(sirius, jobId = character(), all = FALSE)
```

Arguments

sirius	a Sirius object
username	character(1), the username to use for the connection
password	character(1), the password to use for the connection
closeProject	logical, whether to also close the project when closing the GUI. Default is FALSE.
infoType	character vector of length 1 or 2, specifying the type of information to retrieve. Possible values are "compatibilityInfo" and "sizeInformation".
projectId	character(1) specifying the id of the project to open. can be an already existing project or a new one.
path	character(1) path where to find the existing project or where to create a new one. By default, the project will be opened in the current "." directory.
type	character vector of length 1, specifying the type of features ID to list. Possible values are "sirius" and "xcms". Default is "sirius".
featureId	character() vector specifying the id of the feature to remove. By default remove all.
config	a configSirius object
name	character vector of length 1, specifying the name of the configuration to load.
jobId	character(1) specifying the id of the job to delete.
all	logical, whether to delete all jobs. Default is FALSE.

Value

A message whether the user is successfully logged in or not.

logical, TRUE if the connection is valid, FALSE otherwise.

Invisible NULL. Messages indicate shutdown status.

Invisible TRUE if successful.

Invisible TRUE if successful.

a list with the information requested.

a character vector with the open projects.

a Sirius object with the project opened.

a character vector with the features ID (empty if no features).

a data.frame with the features information.

A Sirius object with the features removed.

a data.frame with the features mapping.

nothing, will save the configuration locally.

a character vector with the job information.

nothing, will delete the job from Sirius.

---

zodiacParam *Configuration fr re-ranking of Molecular Formula Annotation*

Description

This function configures the parameters for the re-ranking of the previously computed molecular formula annotation in Sirius. This step is quite computationally and memory demanding, it is advised to perform it only if de novo structure annotation is used later on

ZODIAC uses the top X molecular formula candidates for each molecule from SIRIUS to build a similarity network, and uses Bayesian statistics to re-rank those candidates.

Usage

```
zodiacParam(
  consideredCandidatesAt300Mz = 10,
  consideredCandidatesAt800Mz = 50,
  runInTwoSteps = TRUE,
  edgeFilterThreshold = TRUE,
  thresholdFilter = 0.95,
  minLocalCandidates = 1,
  minLocalConnections = 10,
  gibbsSamplerParameters = TRUE,
  iterations = 20000,
  burnInPeriod = 2000,
  numberOfMarkovChains = 10
)
```

Arguments

consideredCandidatesAt300Mz
: An integer(1) specifying the maximum number of candidate molecular formulas considered by ZODIAC for compounds below 300 m/z. Default is 10.

consideredCandidatesAt800Mz
: An integer(1) specifying the maximum number of candidate molecular formulas considered by ZODIAC for \ compounds above 800 m/z. Default is 50.

`runInTwoSteps` `logical` specifying whether ZODIAC uses a 2-step approach. First running 'good quality compounds' only, and afterwards including the remaining compounds. Default is `TRUE`.

`edgeFilterThreshold`
`logical` value specifying whether ZODIAC uses an edge filter threshold. Default is `TRUE`.

`thresholdFilter`
`numeric(1)` value specifying the threshold filter. Default is `0.95`.

`minLocalCandidates`
`numeric(1)` value specifying the minimum number of local candidates. Default is `1`.

`minLocalConnections`
`numeric(1)` value specifying the minimum number of local connections. Default is `10`.

`gibbsSamplerParameters`
`logical` value specifying whether ZODIAC uses Gibbs sampler parameters. Default is `TRUE`.

`iterations` `numeric(1)` specifying the number of iterations. Default is `20000`.

`burnInPeriod` `numeric(1)` specifying the burn-in period. Default is `2000`.

`numberOfMarkovChains`
`numeric(1)` specifying the number of Markov chains. Default is `10`.

Value

An object of class `zodiacParam`.

Note

For more information, see the Sirius documentation.

References

Ludwig M, Nothias L-F, Dührkop K, et al. (2020). Database-independent molecular formula annotation using Gibbs sampling through ZODIAC. *Nature Machine Intelligence*, 2, 629-641. doi:10.1038/s42256020002346

Examples

```
# Example of setting up the parameter for Zodiac re-ranking
param <- zodiacParam(consideredCandidatesAt300Mz = 10,
                     consideredCandidatesAt800Mz = 50,
                     runInTwoSteps = TRUE,
                     edgeFilterThreshold = TRUE,
                     thresholdFilter = 0.95,
                     minLocalCandidates = 1,
                     minLocalConnections = 10,
                     gibbsSamplerParameters = TRUE,
                     iterations = 20000,
                     burnInPeriod = 2000,
```

```
                numberOfMarkovChains = 10)
```

Index