Package: SpectriPy (via r-universe)

August 31, 2024

Title Integrating Spectra with Python's matchms

Version 0.1.1

Description The SpectriPy package allows integration of Python-based MS analysis code with the Spectra package. Spectra objects can be converted into Python's matchms Spectrum objects. In addition, SpectriPy integrates and wraps the similarity scoring functions from the matchms package into R.

Depends R (>= 4.1.0)

Imports Spectra (>= 1.5.8), basilisk, reticulate, IRanges, S4Vectors, BiocParallel, ProtGenerics, methods

Suggests testthat, knitr, BiocStyle, rmarkdown, patrick

License Artistic-2.0

VignetteBuilder knitr

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

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

biocViews Infrastructure, Metabolomics, MassSpectrometry

Encoding UTF-8

StagedInstall no

SystemRequirements python (>= 3.5)

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

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

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

RemoteRef HEAD

RemoteSha 08c696d22289184f43407d2dbcea2f4190b6630b

Contents

`compareSpectriPy` *Spectra similarity calculations using matchms*

Description

The `compareSpectriPy` function allows to calculate spectral similarity scores using the `calculate_scores module` of the python matchms.similarity package package.

Selection and configuration of the algorithm can be performed with one of the parameter objects:

- `CosineGreedyParam`: calculate the *cosine similarity score* between spectra. The score is calculated by finding best possible matches between peaks of two spectra. Two peaks are considered a potential match if their m/z ratios lie within the given `tolerance`. The underlying peak assignment problem is here solved in a *greedy* way. This can perform notably faster, but does occasionally deviate slightly from a fully correct solution (as with the `CosineHungarianParam` algorithm). In practice this will rarely affect similarity scores notably, in particular for smaller tolerances. The algorithm can be configured with parameters `tolerance`, `mzPower` and `intensityPower` (see parameter description for more details).
- `CosineHungarianParam`: calculate the *cosine similarity score* as with `CosineGreedyParam`, but using the Hungarian algorithm to find the best matching peaks between the compared spectra. The algorithm can be configured with parameters `tolerance`, `mzPower` and `intensityPower` (see parameter description for more details).
- `ModifiedCosineParam`: The modified cosine score aims at quantifying the similarity between two mass spectra. The score is calculated by finding best possible matches between peaks of two spectra. Two peaks are considered a potential match if their m/z ratios lie within the given `tolerance`, or if their m/z ratios lie within the tolerance once a mass-shift is applied. The mass shift is simply the difference in precursor-m/z between the two spectra.

Usage

```
CosineGreedyParam(tolerance = 0.1, mzPower = 0, intensityPower = 1)

CosineHungarianParam(tolerance = 0.1, mzPower = 0, intensityPower = 1)

ModifiedCosineParam(tolerance = 0.1, mzPower = 0, intensityPower = 1)

## S4 method for signature 'Spectra,Spectra,CosineGreedyParam'
compareSpectriPy(x, y, param, ...)

## S4 method for signature 'Spectra,missing,CosineGreedyParam'
compareSpectriPy(x, y, param, ...)
```

Arguments

`tolerance`	`numeric(1)`: tolerated differences in peaks' m/z. Peaks with m/z differences `<= tolerance` are considered matching.

mzPower	numeric(1): the power to raise m/z to in the cosine function. The default is 0, in which case the peak intensity products will not depend on the m/z ratios.
intensityPower	numeric(1): the power to raise intensity to in the cosine function. The default is 1.
x	A Spectra() object.
y	A Spectra() object to compare against. If missing, spectra similarities are calculated between all spectra in x.
param	one of parameter classes listed above (such as CosineGreedyParam) defining the similarity scoring function in python and its parameters.
...	ignored.

Value

compareSpectriPy returns a numeric matrix with the scores, number of rows being equal to length(x) and number of columns equal to length(y).

Author(s)

Carolin Huber, Michael Witting, Johannes Rainer, Helge Hecht

See Also

compareSpectra() in the Spectra package for pure R implementations of spectra similarity calculations.

Examples

```
library(Spectra)
## Create some example Spectra.
DF <- DataFrame(
    msLevel = c(2L, 2L, 2L),
    name = c("Caffeine", "Caffeine", "1-Methylhistidine"),
    precursorMz = c(195.0877, 195.0877, 170.0924)
)
DF$intensity <- list(
    c(340.0, 416, 2580, 412),
    c(388.0, 3270, 85, 54, 10111),
    c(3.407, 47.494, 3.094, 100.0, 13.240))
DF$mz <- list(
    c(135.0432, 138.0632, 163.0375, 195.0880),
    c(110.0710, 138.0655, 138.1057, 138.1742, 195.0864),
    c(109.2, 124.2, 124.5, 170.16, 170.52))
sps <- Spectra(DF)

## Calculate pairwise similarity beween all spectra within sps with
## matchms' CosineGreedy algorithm
## Note: the first compareSpectriPy will take longer because the Python
## environment needs to be set up.
res <- compareSpectriPy(sps, param = CosineGreedyParam())
res
```

```
## Next we calculate similarities for all spectra against the first one
res <- compareSpectriPy(sps, sps[1], param = CosineGreedyParam())

## Calculate pairwise similarity of all spectra in sps with matchms'
## ModifiedCosine algorithm
res <- compareSpectriPy(sps, param = ModifiedCosineParam())
res

## Note that the ModifiedCosine method requires the precursor m/z to be
## known for all input spectra. Thus, it is advisable to remove spectra
## without precursor m/z before using this algorithm.
sps <- sps[!is.na(precursorMz(sps))]
compareSpectriPy(sps, param = ModifiedCosineParam())
```

`rspec_to_pyspec` *Low level functions to convert between Spectra and matchms Spectrum*

Description

The `rspec_to_pyspec` and `pyspec_to_rspec` functions allow to convert R `Spectra()` objects into matchms Python `Spectrum` objects. These functions are designed for **advanced users or developers** who want/need to integrate Python/matchms functionality into R using `reticulate`. All other users should use the dedicated R functions within this package that take care of running the Python code in the correct Python environment.

Parameter `mapping` allows to define which spectra variables (metadata) should be copied between the R and Python spectra. Only provided spectra variables will be copied to R respectively Python. `mapping` also defines the mapping between the `Spectra`'s spectra variables and the Spectrum metadata. The names of the character vector `mapping` are the R spectra variables and the values the corresponding names in the Python's Spectrum metadata. See the output of the `spectraVariableMapping()` function for the default variables and the mapping of the names.

The `spectraVariableMapping` function provides a default mapping of some core `Spectra` variables based on this definition in matchms. The function returns a named vector that can be directly used as parameter `mapping` in the `rspec_to_pyspec` and `pyspec_to_rspec` functions.

Usage

```
## S4 method for signature 'missing'
spectraVariableMapping(object, ...)

rspec_to_pyspec(
  x,
  mapping = spectraVariableMapping(),
  reference = import("matchms"),
  BPPARAM = SerialParam(),
  .check = TRUE
)
```

```
pyspec_to_rspec(
  x,
  mapping = spectraVariableMapping(),
  BPPARAM = SerialParam(),
  .check = TRUE
)
```

Arguments

object	ignored.
...	ignored.
x	For rspec_to_pyspec: Spectra object. For pyspec_to_rspec: an Python list of matchms Spectrum objects.
mapping	Named character providing the spectra variable names (metadata) to convert. Names are expected to be the spectr variable names and values the corresponding names of the Python Spectrum metadata fields. See description above for more details.
reference	Optional reference to Python environment matchms.
BPPARAM	Optional parallel processing setup.
.check	Optionally disable input parameter checking. Input parameter checking should only disabled for very good reasons.

Value

For rspec_to_pyspec: Python array of Spectrum objects, same length than x. Fpr pyspec_to_rspec: Spectra() with the converted spectra. For spectraVariableMapping: named character vector with names being Spectra variable names and values the corresponding names in matchms.

Author(s)

Michael Witting, Johannes Rainer

Examples

```
## List the default spectra variables and their mapping.
spectraVariableMapping()
```

Index