Package 'MSGFplus'

April 12, 2018

Type Package

Title An interface between R and MS-GF+

Version 1.12.0

Date 2017-06-19

Author Thomas Lin Pedersen

Maintainer Thomas Lin Pedersen <thomasp85@gmail.com>

Description This package contains function to perform peptide identification using the MS-GF+ algorithm. The package contains functionality for building up a parameter set both in code and through a simple GUI, as well as running the algorithm in batches, potentially asynchronously.

License GPL (>= 2)

VignetteBuilder knitr

SystemRequirements Java (>= 1.7)

LazyLoad yes

biocViews MassSpectrometry, Proteomics

Depends methods

Imports mzID, ProtGenerics

Suggests gWidgets, knitr, testthat

Collate 'MSGFplus-package.R' 'aaa.R' 'generics.R' 'msgfAsync.R' 'msgfParTolerance.R' 'msgfParTda.R' 'msgfParProtocol.R' 'msgfParNtt.R' 'msgfParModification.R' 'msgfParModificationList.R' 'msgfParMatches.R' 'msgfParLengthRange.R' 'msgfParIsotopeError.R' 'msgfParInstrument.R' 'msgfParFragmentation.R' 'msgfParEnzyme.R' 'msgfParChargeRange.R' 'msgfPar.R' 'msgfPar-getters.R' 'msgfParGUI.R' 'zzz.R'

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

`MSGFplus-package` *An interface between R and MS-GF+*

Description

This package comes bundled with the peptide database search tool MS-GF+ and provides means of running peptide searches from within R with automatic result import and easy handling of parameters.

Details

The version bundled with this package is: MS-GF+ Beta (v10072) (11/11/2013)

Author(s)

Thomas Lin Pedersen with contributions from Laurent Gatto

chargeRange *Get and set the charge range in msgfPar objects*

Description

These functions allow you to retrieve and set the charge range in the msgfPar object of interest

Usage

```
chargeRange(object)

chargeRange(object) <- value

## S4 method for signature 'msgfPar'
chargeRange(object)

## S4 replacement method for signature 'msgfPar,numeric'
chargeRange(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParChargeRange'
chargeRange(object) <- value
```

Arguments

object	An msgfPar object
value	Either a numeric vector of length 2 or an msgfParChargeRange object

Value

In case of the getter a numeric vector with the named elements 'min' and 'max'

Methods (by class)

- msgfPar: Get the charge range
- object = msgfPar,value = numeric: Set the charge range using lower and upper bounds
- object = msgfPar,value = msgfParChargeRange: Set the charge range using a dedicated msgfParChargeRange object

See Also

Other msgfPar-getter_setter: db, enzyme, fragmentation, instrument, isotopeError, lengthRange, matches, mods, ntt, protocol, tda, tolerance

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
chargeRange(parameters) <- c(2, 4)
chargeRange(parameters)
```

db *Get and set database in msgfPar objects*

Description

These functions allow you to retrieve and set the location of the database fasta file in the msgfPar object of interest

Usage

```
db(object)

db(object) <- value

## S4 method for signature 'msgfPar'
db(object)

## S4 replacement method for signature 'msgfPar,character'
db(object) <- value
```

Arguments

object	An msgfPar object
value	A string matching the location of a fasta file

Value

In case of the getter a character vector with the location of the database file

Methods (by class)

- msgfPar: Get the database location
- object = msgfPar,value = character: Set the database location

See Also

Other msgfPar-getter_setter: chargeRange, enzyme, fragmentation, instrument, isotopeError, lengthRange, matches, mods, ntt, protocol, tda, tolerance

Examples

```
parameters <- msgfPar()
db(parameters) <- system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta')
db(parameters)
```

enzyme *Get and set enzyme in msgfPar objects*

Description

These functions allow you to retrieve and set the enzyme used for digestion during sample treatment.

Usage

```
enzyme(object)

enzyme(object) <- value

## S4 method for signature 'msgfPar'
enzyme(object)

## S4 replacement method for signature 'msgfPar,numeric'
enzyme(object) <- value

## S4 replacement method for signature 'msgfPar,character'
enzyme(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParEnzyme'
enzyme(object) <- value
```

Arguments

object	An msgfPar object
value	Either an integer, string or msgfParEnzyme object

Value

In case of the getter a named integer

Methods (by class)

- msgfPar: Get the enzyme currently used
- object = msgfPar,value = numeric: Set the enzyme to use using the key for the enzyme
- object = msgfPar,value = character: Set the enzyme to use using the name of the enzyme
- object = msgfPar,value = msgfParEnzyme: Set the enzyme to use using an msgfParEnzyme object

See Also

Other msgfPar-getter_setter: chargeRange, db, fragmentation, instrument, isotopeError, lengthRange, matches, mods, ntt, protocol, tda, tolerance

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
enzyme(parameters) <- 'Trypsin'
enzyme(parameters) <- 3
enzyme(parameters)
```

`fragmentation` *Get and set fragmentation in msgfPar objects*

Description

These functions allow you to retrieve and set the fragmentation method used during acquisition

Usage

```
fragmentation(object)

fragmentation(object) <- value

## S4 method for signature 'msgfPar'
fragmentation(object)

## S4 replacement method for signature 'msgfPar,numeric'
fragmentation(object) <- value

## S4 replacement method for signature 'msgfPar,character'
fragmentation(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParFragmentation'
fragmentation(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer, string or msgfParFragmentation object

Value

In case of the getter a named integer

Methods (by class)

- `msgfPar`: Get the fragmentation method currently used
- `object = msgfPar,value = numeric`: Set the fragmentation method using the key for the method
- `object = msgfPar,value = character`: Set the fragmentation method using the name of the method
- `object = msgfPar,value = msgfParFragmentation`: Set the fragmentation method using an msgfParFragmentation object

See Also

Other msgfPar-getter_setter: chargeRange, db, enzyme, instrument, isotopeError, lengthRange, matches, mods, ntt, protocol, tda, tolerance

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
fragmentation(parameters) <- 'CID'
fragmentation(parameters) <- 3
fragmentation(parameters)
```

---

getMSGFpar *Get a string representation of an msgfPar-related object*

Description

The string representation is defined as the arguments that should get appended to the call when running MSGF+ in the terminal/command prompt

Usage

```
getMSGFpar(object)
```

Arguments

object	An msgfPar object or a related object

Value

A string that can be appended to a system() call to specify the parameters for the MSGF+ analysis

See Also

msgfPar-class

Examples

```
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C'
                     )
getMSGFpar(parameters)
```

`instrument` *Get and set instrument in msgfPar objects*

Description

These functions allow you to retrieve and set the instrument type used for acquisition

Usage

```
instrument(object)

instrument(object) <- value

## S4 method for signature 'msgfPar'
instrument(object)

## S4 replacement method for signature 'msgfPar,numeric'
instrument(object) <- value

## S4 replacement method for signature 'msgfPar,character'
instrument(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParInstrument'
instrument(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer, string or msgfParInstrument object

Value

In case of the getter a named integer

Methods (by class)

- `msgfPar`: Get the instrument currently used
- `object = msgfPar,value = numeric`: Set the instrument using the key for the instrument
- `object = msgfPar,value = character`: Set the instrument using the name of the instrument
- `object = msgfPar,value = msgfParInstrument`: Set the instrument using an msgfParInstrument object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `isotopeError`, `lengthRange`, `matches`, `mods`, `ntt`, `protocol`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
instrument(parameters) <- 'TOF'
instrument(parameters) <- 3
instrument(parameters)
```

`isotopeError` *Get and set isotope error in msgfPar objects*

Description

These functions allow you to retrieve and set the isotope error used when calculating the parent ion error range

Usage

```
isotopeError(object)

isotopeError(object) <- value

## S4 method for signature 'msgfPar'
isotopeError(object)

## S4 replacement method for signature 'msgfPar,numeric'
isotopeError(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParIsotopeError'
isotopeError(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer vector or an msgfParIsotopeError object

Value

In case of the getter an integer vector

Methods (by class)

- `msgfPar`: Get the isotope error currently used
- `object = msgfPar,value = numeric`: Set the isotope error with an integer vector
- `object = msgfPar,value = msgfParIsotopeError`: Set the isotope error with an msgfParIsotopeError object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `lengthRange`, `matches`, `mods`, `ntt`, `protocol`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
isotopeError(parameters) <- c(0, 3)
isotopeError(parameters)
```

`lengthRange` *Get and set peptide length range in msgfPar objects*

Description

These functions allow you to retrieve and set the residue length allowed for the peptides search for in MS-GF+

Usage

```
lengthRange(object)

lengthRange(object) <- value

## S4 method for signature 'msgfPar'
lengthRange(object)

## S4 replacement method for signature 'msgfPar,numeric'
lengthRange(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParLengthRange'
lengthRange(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer vector or an msgfParLengthRange object

Value

In case of the getter an integer vector of length 2 giving the lower and upper bounds of the length range

Methods (by class)

- `msgfPar`: Get the lower and upper bounds of peptide lengths
- `object = msgfPar,value = numeric`: Set the lower and upper bounds of peptide lengths using an integer vector
- `object = msgfPar,value = msgfParLengthRange`: Set the lower and upper bounds of peptide lengths using an msgfParLengthRange

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `isotopeError`, `matches`, `mods`, `ntt`, `protocol`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
lengthRange(parameters) <- c(6, 40)
lengthRange(parameters)
```

`matches` *Get and set the number of matches in msgfPar objects*

Description

These functions allow you to retrieve and set the number of matches per spectrum returned by MS-GF+

Usage

```
matches(object)

matches(object) <- value

## S4 method for signature 'msgfPar'
matches(object)

## S4 replacement method for signature 'msgfPar,numeric'
matches(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParMatches'
matches(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer or an msgfParMatches object

Value

In case of the getter an integer

Methods (by class)

- `msgfPar`: Get the number of matches reported per spectrum
- `object = msgfPar,value = numeric`: Set the number of matches reported per spectrum using an integer
- `object = msgfPar,value = msgfParMatches`: Set the number of matches reported per spectrum using an msgfParMatches object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `isotopeError`, `lengthRange`, `mods`, `ntt`, `protocol`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
matches(parameters) <- 5
matches(parameters)
```

`mods` *Get and set the modifications in msgfPar objects*

Description

These functions allow you to retrieve and set the specific modifications allowed on peptides during MS-GF+ search, as well as the number allowed on each peptide

Usage

```
mods(object)

mods(object) <- value

nMod(object)

nMod(object) <- value

## S4 method for signature 'msgfPar'
mods(object)

## S4 replacement method for signature 'msgfPar,msgfParModificationList'
mods(object) <- value

## S4 method for signature 'msgfPar'
nMod(object)

## S4 replacement method for signature 'msgfPar,numeric'
nMod(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	An msgfParModificationList object or in the case of nMod an integer

Value

For the getter an msgfParModificationList object or an integer (in the case of nMod)

Methods (by class)

- `msgfPar`: Get the list of modifications allowed during peptide search
- `object = msgfPar,value = msgfParModificationList`: Set the list of modifications allowed during peptide search
- `msgfPar`: Get the number of peptides allowed per peptide during search
- `object = msgfPar,value = numeric`: Set the number of peptides allowed per peptide during search using an integer

See Also

Other msgfPar-getter_setter: chargeRange, db, enzyme, fragmentation, instrument, isotopeError, lengthRange, matches, ntt, protocol, tda, tolerance

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
nMod(parameters) <- 2
mods(parameters)[[1]] <- msgfParModification(
                                                    name='Carbamidomethyl',
                                                    composition='C2H3N1O1',
                                                    residues='C',
                                                    type='fix',
                                                    position='any'
                                                   )
mods(parameters)
```

`msgfPar` *Constructor for the msgfPar class*

Description

This function creates an msgfPar object with the specified parameters. If some parameters have not been specified they will not be part of the MS-GF+ call and MS-GF+'s own defaults kicks in; Consult the MS-GF+ documentation for these. Note however that at least a database file is required to run an analysis.

Usage

```
msgfPar(database, tolerance, isotopeError, tda, fragmentation, instrument,
  enzyme, protocol, ntt, modification, lengthRange, chargeRange, matches)
```

Arguments

`database`	The location of the fasta file to use as search database
`tolerance`	The parent ion tolerance to use. In simple cases a string in the form '20 ppm' or '1 Da' or an msgfParTolerance object if asymmetric tolerance is desired
`isotopeError`	The range of isotope errors used to correct for non-monoisotopic peaks. Either a numeric vector of length 2 specifying the lower and upper bounds of the range, or an msgfParIsotopeError object
`tda`	`Logical` Should Target-Decoy approach be used to calculate FDR values.
`fragmentation`	An integer specifying which fragmentation has been used during data acquisition. See details.
`instrument`	An integer specifying the type of instrument used during data acquisition. See details.
`enzyme`	An integer or name specifying the enzyme that has been used for protein digestion. See details.
`protocol`	An integer or name specifying the type of preparation that has been done for the samples. See details.

`ntt`	An integer specifying the cleavage specificity (Number of Tolerable Termini). 2 only allows fully tryptic peptide (if trypsin is used), 1 allows semitryptic peptides and 0 allows unspecific peptides
`modification`	An msgfParModificationList object or a list containing the named elements nMod and modifications containing respectively an integer with the number of allowed modifications per petide and the modifications to search for as msgfParModification
`lengthRange`	A two element vector containing the lower and upper bounds of the residue length to search for
`chargeRange`	A two element vector containing the lower and upper bounds of the charge range to search for
`matches`	The number of matches to report per spectrum

Details

Please consult the MS-GF+ documentation for full description of the parameters

Fragmentation is usually specified as an integer according to the following lookup

0 As written in the spectrum or CID if no info
1 CID
2 ETD
3 HCD
4 Merge spectra from the same precursor

It is possible to use the full name of the description for a more litteral function call

Instrument can likewise be specified as an integer or as a name according to this list

0 LowRes
1 HighRes
2 TOF
3 QExactive

Enymes are specified in the same manner using the following list

0 Unspecific cleavage
1 Trypsin
2 Chymotrypsin
3 Lys-C
4 Lys-N
5 glutamyl endopeptidase (Glu-C)
6 Arg-C
7 Asp-N
8 alphaLP
9 No cleavage

The protocol informs MS-GF+ whether a special sample treatment has been performed as part of the analysis. The protocol is specified according to the following list

0 No protocol
1 Phosphorylation
2 iTRAQ
3 iTRAQPhospho

Value

An msgfPar object

References

MS-GF+

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class

Examples

```
# Example of specifying all parameters - usually not necessary
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      isotopeError=c(0, 2),
                      tda=TRUE,
                      fragmentation='CID',
                      instrument='TOF',
                      enzyme='Lys-C',
                      protocol='No protocol',
                      ntt=2,
                      modification=list(
                          nMod=2,
                          modifications=list(
                              list(name='Carbamidomethyl',
                                   composition='C2H3N1O1',
                                   residues='C',
                                   type='fix',
                                   position='any'
                              ),
                              list(name='Oxidation',
                                   mass=15.994915,
                                   residues='M',
                                   type='opt',
                                   position='any'
                              )
                          )
                      ),
                      lengthRange=c(6,40),
                      chargeRange=c(2,7),
                      matches=1
                     )
parameters
```

msgfPar-class *A class to contain parameters used in an MS-GF+ analysis*

Description

This class collects and stores parameters for an MS-GF+ analysis and is the starting point for peptide identification

Usage

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

## S4 method for signature 'msgfPar'
length(x)

## S4 method for signature 'msgfPar'
getMSGFpar(object)

## S4 method for signature 'msgfPar'
runMSGF(object, rawfiles, savenames, import = TRUE,
  memory = 10000, async = FALSE, msgfPath)
```

Arguments

object	An msgfPar object
x	An msgfPar object
rawfiles	A character vector holding the filepath to the spectrum files to be analysed (currently supported formats: *.mzML, *.mzXML, *.mgf, *.ms2, *.pkl or *_dta.txt)
savenames	An optinal vector of same length as rawfiles. Specifies the name used to save the results. If omitted the results will be saved with the same name as the rawfile, but with an .mzid extension.
import	Logical (default=TRUE). Should the results be imported in to R after the analysis is finished.
memory	An integer (default=10000). How much memory should be allocated to the java virtual machine during execution (in mb)
async	An Logical (default=FALSE). Should MS-GF+ be run asynchronously?
msgfPath	The path to an alternative MSGFPlus.jar file if the bundled one is not desired

Details

This class contains a range of other classes, each handling a different set of parameters. Often these classes are simple containers that only takes care of errorchecking and generating command line arguments, but in some cases, as with msgfParModificationList, the class is a bit more complex.

Value

`length`: 1 if a database is defined, 0 otherwise.

`getMSGFpar`: A stringified version of the parameters compliant with MS-GF+.

`runMSGF`: If `import=TRUE` an mzID or mzIDCollection object. If `async=TRUE` an msgfAsync object. Otherwise NULL

Methods (by generic)

- `show`: Short summary of msgfPar object
- `length`: Report the length of an msgfPar object
- `getMSGFpar`: Get `system` compliant function call
- `runMSGF`: Initiate an MS-GF+ analysis using the selected msgfPar object

Slots

`database` The location of the database fasta file used for the analysis.

`tolerance` An `msgfParTolerance` object holding the m/z tolerance used in the search.

`isotopeError` An `msgfParIsotopeError` object holding the isotope errors permitted in the search.

`tda` An `msgfParTda` object saying whether FDR should be estimated using the target-decoy approach.

`fragmentation` An `msgfParFragmentation` object holding the type of fragmentation expected from the experiment.

`instrument` An `msgfParInstrument` object holding which type of instrument was used for collecting the data.

`enzyme` An `msgfParEnzyme` object holding which enzyme was used for digestion

`protocol` An `msgfParProtocol` object defining whether a specific protocol should be used in the search.

`ntt` An `msgfParNtt` object defining the number of tolerable termini allowed in the peptides.

`modification` An `msgfParModificationList` object holding the modifications accepted in the search.

`lengthRange` An `msgfParLengthRange` object setting the limits on the peptide length in residues that the search allows.

`chargeRange` An `msgfParChargeRange` object defining which charges should be included in the search.

`matches` An `msgfParMatches` object defining the number of matches per PSM that gets reported in the output.

Objects from the class

Objects can be created using the `msgfPar` constructor, or with `msgfParGUI` for a simple graphical user interface

References

http://proteomics.ucsd.edu/Software/MSGFPlus.html

See Also

msgfParGUI

Other msgfParClasses: msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C'
                     )
getMSGFpar(parameters)

## Not run:
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C'
                     )
runMSGF(parameters, c('file1.mzML', 'file2.mzML'))

## End(Not run)
```

---

msgfParChargeRange-class

A class handling charge ranges

Description

This class defines a charge range and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParChargeRange'
length(x)

## S4 method for signature 'msgfParChargeRange'
getMSGFpar(object)

msgfParChargeRange(value)
```

Arguments

object	An msgfParChargeRange object
x	An msgfParChargeRange object
value	A numeric vector of length 2. The first element must be smaller than the last

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParChargeRange() An msgfParChargeRange object.

Methods (by generic)

- show: Short summary of msgfParChargeRange object
- length: Report the length of an msgfParChargeRange object
- getMSGFpar: Get system compliant function call

Slots

value A numeric vector of length 2 describing the upper and lower bounds of the charge range

See Also

Other msgfParClasses: msgfPar-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
charges <- msgfParChargeRange(c(2, 5))
```

---

msgfParEnzyme-class *A class handling enzyme selection*

Description

This class defines a digestion enzyme selection and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParEnzyme'
length(x)

## S4 method for signature 'msgfParEnzyme'
```

```
getMSGFpar(object)

msgfParEnzyme(enzyme)
```

Arguments

object	An msgfParEnzyme object
x	An msgfParEnzyme object
enzyme	Either an integer or a string

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParEnzyme() An msgfParEnzyme object.

Methods (by generic)

- show: Short summary of msgfParEnzyme object
- length: Report the length of an msgfParEnzyme object
- getMSGFpar: Get system compliant function call

Slots

enzyme An integer specifiying the selection of enzyme. See the detail section of msgfPar

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
enzyme <- msgfParEnzyme(1)
enzyme <- msgfParEnzyme('Trypsin')
```

---

msgfParFragmentation-class

A class handling Fragmentation types

Description

This class defines a fragmentation type and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParFragmentation'
length(x)

## S4 method for signature 'msgfParFragmentation'
getMSGFpar(object)

msgfParFragmentation(method)
```

Arguments

object	An msgfParFragmentation object
x	An msgfParFragmentation object
method	An integer specifying the method

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParFragmentation() An msgfParFragmentation object.

Methods (by generic)

- show: Short summary of msgfParFragmentation object
- length: Report the length of an msgfParFragmentation object
- getMSGFpar: Get system compliant function call

Slots

method An integer between 0 and 4 giving the selected method

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
fragmentation <- msgfParFragmentation(1)
fragmentation <- msgfParFragmentation('CID')
```

msgfParFromID *Extract parameters from mzIdentML result file*

Description

This function analyses an mzIdentML file generated using MS-GF+ and returns an msgfPar object with parameters matching the ones used to generate the mzIdentML file. If the mzIdentML file does not origin from an MS-GF+ analysis it throws an error.

Usage

```
msgfParFromID(file)
```

Arguments

file	The mzIdentML file to extract the parameters from

Details

NOTE: At the moment the number of allowed modifications per peptide is not written to the result file and can thus not be extracted. It defaults to 2

Value

An msgfPar object with parameters matching the input file

See Also

msgfPar-class msgfPar

Examples

```
## Not run:
parameters <- msgfParFromID('result1.mzid')

## End(Not run)
```

msgfParGUI *A simple GUI to create msgfPar objects*

Description

This function presents the user with a GUI where the different parameters can be filled out interactively. When the window appears the different values already present reflects the default values for MS-GF+ so leaving them as is equals to not setting them in advance.

Usage

```
msgfParGUI()
```

Details

NOTE: This functions requires gWidgets and checks for the existance beforehand. MSGFplus does not import gWidgets and gWidgets does thus not necessarily exist on your system. In addition at least one of the gWidgetsXXX packages are needed.

Value

A msgfPar object with parameters set according to the final state of the GUI

See Also

msgfPar-class

Examples

```
## Not run:
parameters <- msgfParGUI()

## End(Not run)
```

---

msgfParInstrument-class

A class handling instrument types

Description

This class defines an instrument type and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParInstrument'
length(x)

## S4 method for signature 'msgfParInstrument'
getMSGFpar(object)

msgfParInstrument(instrument)
```

Arguments

object	An msgfParInstrument object
x	An msgfParInstrument object
instrument	An integer specifying the instrument type

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParInstrument() An msgfParInstrument object.

Methods (by generic)

- show: Short summary of msgfParInstrument object
- length: Report the length of an msgfParInstrument object
- getMSGFpar: Get system compliant function call

Slots

instrument An integer specifying the instrument type

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-cla msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
instrument <- msgfParInstrument(1)
instrument <- msgfParInstrument('HighRes')
```

---

msgfParIsotopeError-class

A class handling isotope errors

Description

This class defines a set of isotopes that should be included for error correction and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParIsotopeError'
length(x)

## S4 method for signature 'msgfParIsotopeError'
getMSGFpar(object)

msgfParIsotopeError(range)
```

Arguments

object	An msgfParIsotopeError object
x	An msgfParIsotopeError object
range	An integer vector with isotopes

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParIsotopeError() An msgfParIsotopeError object.

Methods (by generic)

- show: Short summary of msgfParIsotopeError object
- length: Report the length of an msgfParIsotopeError object
- getMSGFpar: Get system compliant function call

Slots

range An integer vector with lower and upper bounds of isotopes to error correct

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParLengthRange-class, msgfParMatches-class msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
isotopeError <- msgfParIsotopeError(c(0, 2))
```

---

msgfParLengthRange-class

A class handling length ranges

Description

This class defines a length range and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParLengthRange'
length(x)

## S4 method for signature 'msgfParLengthRange'
```

```
getMSGFpar(object)

msgfParLengthRange(value)
```

Arguments

object	An msgfParLengthRange object
x	An msgfParLengthRange object
value	A numeric vector of length 2. The first element must be smaller than the last

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParLengthRange() An msgfParLengthRange object.

Methods (by generic)

- show: Short summary of msgfParLengthRange object
- length: Report the length of an msgfParLengthRange object
- getMSGFpar: Get system compliant function call

Slots

value A numeric vector of length 2 describing the upper and lower bounds of the length range

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParMatches-clas msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
lengths <- msgfParLengthRange(c(6, 40))
```

---

msgfParMatches-class *A class handling number of matches*

Description

This class defines a number of matches and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParMatches'
length(x)

## S4 method for signature 'msgfParMatches'
getMSGFpar(object)

msgfParMatches(value)
```

Arguments

object	An msgfParMatches object
x	An msgfParMatches object
value	An integer giving the number of matches that should be returned per spectrum

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParMatches() An msgfParMatches object.

Methods (by generic)

- show: Short summary of msgfParMatches object
- length: Report the length of an msgfParMatches object
- getMSGFpar: Get system compliant function call

Slots

value An integer giving the number of matches per spectrum reported by MS-GF+

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
matches <- msgfParMatches(5)
```

`msgfParModification-class`
A class handling modification characteristics

Description

This class defines a single modification type that can be searched for by MS-GF+. Modifications are collected in a `msgfParModificationList` before adding them to `msgfPar` objects.

Usage

```
## S4 method for signature 'msgfParModification'
getMSGFpar(object)

## S4 method for signature 'msgfParModification'
show(object)

msgfParModification(name, composition = "", mass = as.numeric(NA), residues,
  type, position)
```

Arguments

`object`	An msgfParModification object
`name`	The name of the modification
`composition`	The molecular formular as a string for the modification. Loss of atoms are denoted with negative integers (e.g. O-1 for loss of oxygen)
`mass`	The monoisotopic mass change between a peptide without and with the given modification. Either composition or mass must be defined.
`residues`	The amino acids that the modification applies to. Given as their one-letter code in upper-case without any separation. Use '*' for all residues
`type`	Either 'fix' or 'opt' for fixed or optional
`position`	Where the modification can be. Either 'any', 'c-term', 'n-term', 'prot-c-term' or 'prot-n-term.

Value

For getMSGFpar() A string.

For msgfParModification() An msgfParModification object.

Methods (by generic)

- `getMSGFpar`: Get `system` compliant function call
- `show`: Short summary of msgfParModification object

Slots

`composition` The molecular formula for the modification.

`mass` The monoisotopic mass of the modification

`residues` The amino acids the modification applies to

`type` Whether the modification is optional or always present

`position` The possibel position of the modification

`name` The name of the modification

See Also

Other msgfParClasses: `msgfPar-class`, `msgfParChargeRange-class`, `msgfParEnzyme-class`, `msgfParFragmentation-class`, `msgfParInstrument-class`, `msgfParIsotopeError-class`, `msgfParLengthRange-class`, `msgfParMatches-class`, `msgfParModificationList-class`, `msgfParNtt-class`, `msgfParProtocol-class`, `msgfParTda-class`, `msgfParTolerance-class`, `msgfPar`

Examples

```
# Using composition
modification1 <- msgfParModification(
                                     name='Carbamidomethyl',
                                     composition='C2H3N1O1',
                                     residues='C',
                                     type='fix',
                                     position='any'
                                    )
# Using exact mass
modification2 <- msgfParModification(
                                     name='Oxidation',
                                     mass=15.994915,
                                     residues='M',
                                     type='opt',
                                     position='any'
                                    )
```

---

msgfParModificationList-class

A class handling a list of modifications

Description

This class defines a set of modifications and a maximum number of modifications allowed per peptide.

Usage

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

## S4 method for signature 'msgfParModificationList'
length(x)
```

```
## S4 method for signature 'msgfParModificationList'
getMSGFpar(object)

## S4 method for signature 'msgfParModificationList,numeric,missing'
x[[i, j, ...]]

## S4 replacement method for signature
## 'msgfParModificationList,numeric,missing,msgfParModification'
x[[i,
  j, ...]] <- value

msgfParModificationList(nMod, modifications = list())
```

Arguments

object	An msgfParModificationList object
x	An msgfParModificationList object
i	The index of the modification
j	Ignored
...	Ignored
value	An msgfParModification object
nMod	The maximum allowed number of modifications to expect on any peptide
modifications	A list of msgfParModification objects

Value

For length() An integer.

For getMSGFpar() A string.

For '[[' A msgfParModification object

For msgfParModificationList() An msgfParModificationList object.

Methods (by generic)

- show: Short summary of msgfParModificationList object
- length: Report the length of an msgfParModificationList object
- getMSGFpar: Get system compliant function call
- [[: Get the i'th modification
- [[<-: Set or change the i'th modification

Slots

nMod The maximum allowed number of modifications to expect on any peptide

modifications A list of msgfParModification objects

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-msgfParMatches-class, msgfParModification-class, msgfParNtt-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
modification1 <- msgfParModification(
                                     name='Carbamidomethyl',
                                     composition='C2H3N1O1',
                                     residues='C',
                                     type='fix',
                                     position='any'
                                    )
modification2 <- msgfParModification(
                                     name='Oxidation',
                                     mass=15.994915,
                                     residues='M',
                                     type='opt',
                                     position='any'
                                    )
modificationlist <- msgfParModificationList(
                                            nMod=2,
                                            modifications=list(
                                                modification1,
                                                modification2
                                            )
                                           )
modificationlist[[3]] <- msgfParModification(
                                             name='Gln->pyro-Glu',
                                             composition='H-3N-1',
                                             residues='Q',
                                             type='opt',
                                             position='N-term'
                                            )
```

`msgfParNtt-class` *A class handling cleavage specificity*

Description

This class defines cleavage specificity and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParNtt'
length(x)

## S4 method for signature 'msgfParNtt'
getMSGFpar(object)

msgfParNtt(value)
```

Arguments

object	An msgfParNtt object
x	An msgfParNtt object
value	An integer between 0 and 2 that specifies the specificity (2: full cleavage, 1: semi specific cleavage, 0: no specificity)

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParNtt() An msgfParNtt object.

Methods (by generic)

- show: Short summary of msgfParNtt object
- length: Report the length of an msgfParNtt object
- getMSGFpar: Get system compliant function call

Slots

value An integer between 0 and 2 that specifies the specificity (2: full cleavage, 1: semi specific cleavage, 0: no specificity)

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParProtocol-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
ntt <- msgfParNtt(2)
```

---

msgfParProtocol-class *A class handling protocol choice*

Description

This class defines a protocol and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParProtocol'
length(x)

## S4 method for signature 'msgfParProtocol'
getMSGFpar(object)

msgfParProtocol(protocol)
```

Arguments

object	An msgfParProtocol object
x	An msgfParProtocol object
protocol	An integer or string specifying the protocol to use.

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParProtocol() An msgfParProtocol object.

Methods (by generic)

- show: Short summary of msgfParProtocol object
- length: Report the length of an msgfParProtocol object
- getMSGFpar: Get system compliant function call

Slots

protocol An integer specifying a specific protocol type

See Also

Other msgfParClasses: msgfPar-class, msgfParChargeRange-class, msgfParEnzyme-class, msgfParFragmentation-class, msgfParInstrument-class, msgfParIsotopeError-class, msgfParLengthRange-class, msgfParMatches-class, msgfParModification-class, msgfParModificationList-class, msgfParNtt-class, msgfParTda-class, msgfParTolerance-class, msgfPar

Examples

```
protocol <- msgfParProtocol(0)
protocol <- msgfParProtocol('No protocol')
```

`msgfParTda-class` *A class handling use of target-decoy approach for FDR estimation*

Description

This class defines whether to use target-decoy approach and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParTda'
length(x)

## S4 method for signature 'msgfParTda'
getMSGFpar(object)

msgfParTda(value)
```

Arguments

`object`	An msgfParTda object
`x`	An msgfParTda object
`value`	A boolean defining whether to use tda or not

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParTda() An msgfParTda object.

Methods (by generic)

- `show`: Short summary of msgfParTda object
- `length`: Report the length of an msgfParTda object
- `getMSGFpar`: Get `system` compliant function call

Slots

`tda` A boolean defining whether to use tda or not

See Also

Other msgfParClasses: `msgfPar-class`, `msgfParChargeRange-class`, `msgfParEnzyme-class`, `msgfParFragmentation-class`, `msgfParInstrument-class`, `msgfParIsotopeError-class`, `msgfParLengthRange-msgfParMatches-class`, `msgfParModification-class`, `msgfParModificationList-class`, `msgfParNtt-class`, `msgfParProtocol-class`, `msgfParTolerance-class`, `msgfPar`

Examples

```
tda <- msgfParTda(TRUE)
```

`msgfParTolerance-class` *A class handling parent ion tolerance*

Description

This class defines a parent ion tolerance and provides methods to get correct system call parameters.

Usage

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

## S4 method for signature 'msgfParTolerance'
length(x)

## S4 method for signature 'msgfParTolerance'
getMSGFpar(object)

msgfParTolerance(value, low, high, unit)
```

Arguments

`object`	An msgfParTolerance object
`x`	An msgfParTolerance object
`value`	A numeric giving the upper and lower bounds of the tolerance
`low`	A numeric giving the lower bounds of the tolerance
`high`	A numeric giving the higher bounds of the tolerance
`unit`	The unit used. Either 'ppm' or 'Da'

Value

For length() An integer.

For getMSGFpar() A string.

For msgfParTolerance() An msgfParTolerance object.

Methods (by generic)

- `show`: Short summary of msgfParTolerance object
- `length`: Report the length of an msgfParTolerance object
- `getMSGFpar`: Get `system` compliant function call

Slots

`unit` The unit used to define the tolerance

`low` The lower bound of the tolerance

`high` The higher bound of the tolerance

See Also

Other msgfParClasses: `msgfPar-class`, `msgfParChargeRange-class`, `msgfParEnzyme-class`, `msgfParFragmentation-class`, `msgfParInstrument-class`, `msgfParIsotopeError-class`, `msgfParLengthRange-`
`msgfParMatches-class`, `msgfParModification-class`, `msgfParModificationList-class`, `msgfParNtt-class`, `msgfParProtocol-class`, `msgfParTda-class`, `msgfPar`

Examples

```
# Symmetric
tolerance <- msgfParTolerance(20, unit='ppm')

# Asymmetric
tolerance <- msgfParTolerance(low=0.5, high=1.5, unit='Da')
```

---

ntt *Get and set cleavage specificity in msgfPar objects*

Description

These functions allow you to retrieve and set the quality of cleavage allowed during search in MS-GF+ (number of tolerable termini - ntt)

Usage

```
ntt(object)

ntt(object) <- value

## S4 method for signature 'msgfPar'
ntt(object)

## S4 replacement method for signature 'msgfPar,numeric'
ntt(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParNtt'
ntt(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	An integer or an msgfParNtt object

Value

In case of the getter an integer between 0 and 2

Methods (by class)

- `msgfPar`: Get the number of tolerable termini
- `object = msgfPar,value = numeric`: Set the ntt using an integer
- `object = msgfPar,value = msgfParNtt`: Set the ntt using an msgfParNtt object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `isotopeError`, `lengthRange`, `matches`, `mods`, `protocol`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
ntt(parameters) <- 2
ntt(parameters)
```

---

protocol *Get and set protocol in msgfPar objects*

Description

These functions allow you to retrieve and set the protocol used during MS-GF+ analysis. This allows you to fine tune the analysis in case of labelled or phosphoproteomic analysis

Usage

```
protocol(object)

protocol(object) <- value

## S4 method for signature 'msgfPar'
protocol(object)

## S4 replacement method for signature 'msgfPar,numeric'
protocol(object) <- value

## S4 replacement method for signature 'msgfPar,character'
protocol(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParProtocol'
protocol(object) <- value
```

Arguments

`object`	An msgfPar object
`value`	Either an integer, a string or an msgfParProtocol object

Value

In case of the getter a named integer

Methods (by class)

- `msgfPar`: Get the protocol currently used
- `object = msgfPar,value = numeric`: Set the protocol using the key for the protocol
- `object = msgfPar,value = character`: Set the protocol using the name of the protocol
- `object = msgfPar,value = msgfParProtocol`: Set the protocol using an msgfParProtocol object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `isotopeError`, `lengthRange`, `matches`, `mods`, `ntt`, `tda`, `tolerance`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
protocol(parameters) <- 'Phosphorylation'
protocol(parameters) <- 0
protocol(parameters)
```

---

runMSGF *Runs MS-GF+ based on the given msgfPar object*

Description

This function assembles a system call based on the parameters specified in the object and the arguments given in the function call. By default the function uses the MS-GF+ jar file bundled with this package, but it is possible to specify an alternative location using the msgfPar argument. Version compatibility can not be assured in this case though.

Usage

```
runMSGF(object, rawfiles, savenames, import, memory, async, msgfPath)
```

Arguments

`object`	An msgfPar object
`rawfiles`	A character vector holding the filepath to the spectrum files to be analysed (currently supported formats: *.mzML, *.mzXML, *.mgf, *.ms2, *.pkl or *_dta.txt)
`savenames`	An optinal vector of same length as rawfiles. Specifies the name used to save the results. If omitted the results will be saved with the same name as the rawfile, but with an .mzid extension.
`import`	Logical (default=TRUE). Should the results be imported in to R after the analysis is finished.
`memory`	An integer (default=10000). How much memory should be allocated to the java virtual machine during execution (in mb)
`async`	An Logical (default=FALSE). Should MS-GF+ be run asynchronously?
`msgfPath`	The path to an alternative MSGFPlus.jar file if the bundled one is not desired

Value

If import=TRUE a list of mzID object otherwise NULL

See Also

mzID

Examples

```
## Not run:
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C'
                     )
runMSGF(parameters, c('file1.mzML', 'file2.mzML'))

## End(Not run)
```

---

running *A class referencing an asynchronous execution of MS-GF+*

Description

Objects of this class contains a reference to an asynchronous running MS-GF+ process and can be used to query the state of the process, and import the results if the process has finished. Instances of this class are created when the runMSGF() method is called with `async=TRUE` on an msgfPar object.

Unlike regular runMSGF() calls, this does not support batch mode, meaning that if a more than one raw file is supplied, all but the first are ignored with a warning.

In order to insulate instances of this class from being corrupted (thus loosing the reference to the process), all slots are functions and should be queried as such if needed.

Usage

```
running(object)

finished(object)

import(object)

## S4 method for signature 'msgfAsync'
running(object)

## S4 method for signature 'msgfAsync'
finished(object)

## S4 method for signature 'msgfAsync'
import(object)
```

Arguments

object	An msgfAsync object

Value

`running(object)` Returns a logical indicating if the process is running

`finished(object)` Returns a logical indicating if the process is finished

`import(object)` Returns an mzID object or NULL if the process is still running. Throws an error if the process is finished but the result file doesn't exist.

Methods (by generic)

- `running`: Check whether the MS-GF+ process is still running
- `finished`: Check whether the MS-GF+ process is finished
- `import`: Import the result of the asynchronous MS-GF+ process

Slots

`status` Returns the status of the MS-GF+ process; either 'Running' or 'Done'.

`resultFile` Returns the location of the result file from the MS-GF+ analysis. WARNING: Checking for the existence of this file is not a safe way to determine the status of the process, as the file gets written to continuously.

Examples

```
## Not run:
parameters <- msgfPar(
                  database=system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'),
                      tolerance='20 ppm',
                      instrument='TOF',
                      enzyme='Lys-C'
                     )
asyncMSGF <- runMSGF(parameters, 'file1.mzML', async=TRUE)
while(!running(asyncMSGF)){
    Sys.sleep(1)
}
results <- import(asyncMSGF)

## End(Not run)
```

---

tda *Get and set use of target-decoy approach in msgfPar objects*

Description

These functions allow you to retrieve and set whether the target-decoy approach should be used to estimate q-values.

Usage

```
tda(object)

tda(object) <- value

## S4 method for signature 'msgfPar'
tda(object)

## S4 replacement method for signature 'msgfPar,logical'
tda(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParTda'
tda(object) <- value
```

Arguments

object	An msgfPar object
value	Either a boolean or msgfParTda object

Value

In case of the getter a boolean indicating whether tda is used or not

Methods (by class)

- msgfPar: Get whether tda is currently used for FDR estimation
- object = msgfPar,value = logical: Set the use of tda using a boolean (TRUE/FALSE)
- object = msgfPar,value = msgfParTda: Set the use of tda using an msgfParTda object

See Also

Other msgfPar-getter_setter: chargeRange, db, enzyme, fragmentation, instrument, isotopeError, lengthRange, matches, mods, ntt, protocol, tolerance

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
tda(parameters) <- TRUE
tda(parameters)
```

---

tolerance *Get and set the parent tolerance in msgfPar objects*

Description

These functions allow you to retrieve and set the tolerance used for matching parent ions to peptides in the database

Usage

```
tolerance(object)

tolerance(object) <- value

toleranceRange(object)

toleranceRange(object) <- value

toleranceUnit(object)

toleranceUnit(object) <- value

## S4 method for signature 'msgfPar'
tolerance(object)

## S4 method for signature 'msgfPar'
toleranceRange(object)

## S4 method for signature 'msgfPar'
toleranceUnit(object)

## S4 replacement method for signature 'msgfPar,numeric'
toleranceRange(object) <- value

## S4 replacement method for signature 'msgfPar,character'
toleranceUnit(object) <- value

## S4 replacement method for signature 'msgfPar,character'
tolerance(object) <- value

## S4 replacement method for signature 'msgfPar,msgfParTolerance'
tolerance(object) <- value
```

Arguments

object	An msgfPar object
value	For tolerance a character vector of length 2, each element of the form '<value> <unit>'. For toleranceUnit a string. For toleranceRange a numeric vector of length 2.

Value

For tolerance a character vector with the lower and upper tolerance limit with unit. For toleranceUnit a string with the unit used. For toleranceRange a numeric vector with lower and upper tolerance limit.

Methods (by class)

- msgfPar: Get the lower and upper bounds of the tolerance
- msgfPar: Get the lower and upper bounds of the tolerance
- msgfPar: Get the unit the tolerance is measured in

- `object = msgfPar,value = numeric`: Set the lower and upper bounds of the tolerance using a numeric vector of length 2
- `object = msgfPar,value = character`: Set the unit the tolerance is meassured in
- `object = msgfPar,value = character`: Set the lower and upper bounds of the tolerance using a character vector of length 2, where each element is of the form '<value> <unit>'
- `object = msgfPar,value = msgfParTolerance`: Set the lower and upper bounds of the tolerance using an msgfParTolerance object

See Also

Other msgfPar-getter_setter: `chargeRange`, `db`, `enzyme`, `fragmentation`, `instrument`, `isotopeError`, `lengthRange`, `matches`, `mods`, `ntt`, `protocol`, `tda`

Examples

```
parameters <- msgfPar(system.file(package='MSGFplus', 'extdata', 'milk-proteins.fasta'))
tolerance(parameters) <- c('20 ppm', '20 ppm')
toleranceUnit(parameters) <- 'Da'
toleranceRange(parameters) <- c(1.5, 1.5)
tolerance(parameters)
```

Index