

Package ‘topdownr’

April 12, 2018

Title Investigation of Fragmentation Conditions in Top-Down Proteomics

Version 1.0.0

Description The topdownr package allows automatic and systemic investigation of fragment conditions. It creates Thermo Orbitrap Fusion Lumos method files to test hundreds of fragmentation conditions. Additionally it provides functions to analyse and process the generated MS data and determine the best conditions to maximise overall fragment coverage.

Depends R (>= 3.4), methods, BiocGenerics (>= 0.20.0), Biostrings (>= 2.42.1), S4Vectors (>= 0.12.2)

Imports stats, tools, utils, Biobase, Matrix (>= 1.2.10), MSnbase (>= 2.3.10), ggplot2 (>= 2.2.1), mzR (>= 2.11.4)

Suggests topdownrdata (>= 0.2), knitr, ranger, testthat, BiocStyle

License GPL (>= 3)

URL <https://github.com/sgibb/topdownr/>

BugReports <https://github.com/sgibb/topdownr/issues/>

LazyData true

VignetteBuilder knitr

Roxygen list(markdown=TRUE)

RoxygenNote 6.0.1.9000

biocViews Infrastructure, Proteomics, MassSpectrometry, Coverage

Encoding UTF-8

NeedsCompilation no

Author Sebastian Gibb [aut, cre] (0000-0001-7406-4443),
Pavel Shliaha [aut],
Ole Nørregaard Jensen [aut]

Maintainer Sebastian Gibb <mail@sebastiangibb.de>

R topics documented:

topdownr-package	2
AbstractTopDownSet-class	3
createTngFusionMethFiles	6
defaultMs1Settings	8

FragmentViews-class	8
NCBSet-class	10
readTopDownFiles	12
tds	14
TopDownSet-class	14
writeMethodXmIs	18
Index	20

topdownr-package	<i>Investigation of Fragmentation Conditions in Top-Down Proteomics</i>
------------------	---

Description

The topdownr package allows automatic and systemic investigation of fragment conditions. It creates Thermo Orbitrap Fusion Lumos method files to test hundreds of fragmentation conditions. Additionally it provides functions to analyse and process the generated MS data and determine the best conditions to maximise overall fragment coverage.

Details

The usage of the topdownr package is demonstrated in the following vignettes:

- Generate .meth files prior data acquisition for the Thermo Orbitrap Fusion Lumos MS devise: `vignette("data-generation", package="topdownr")`.
- How to analyse top-down fragmentation data: `vignette("analysis", package="topdownr")`

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>, Pavel Shliaha <pavels@bmb.sdu.dk>, Ole Nørregaard Jensen <jenseno@bmb.sdu.dk>

References

<https://github.com/sgibb/topdownr/>

See Also

Useful links:

- <https://github.com/sgibb/topdownr/>
- Report bugs at <https://github.com/sgibb/topdownr/issues/>

AbstractTopDownSet-class

The AbstractTopDownSet class

Description

Abstract/VIRTUAL parent class for [TopDownSet](#) and [NCBSet](#) to provide common interface.

Usage

```
## S4 method for signature 'AbstractTopDownSet,ANY,ANY,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'AbstractTopDownSet,ANY,missing'
x[[i, j, ...]]

## S4 replacement method for signature 'AbstractTopDownSet,ANY,missing'
x[[i, j, ...]] <- value

## S4 method for signature 'AbstractTopDownSet'
x$name

## S4 replacement method for signature 'AbstractTopDownSet'
x$name <- value

## S4 method for signature 'AbstractTopDownSet'
assayData(object)

## S4 method for signature 'AbstractTopDownSet'
colData(object)

## S4 replacement method for signature 'AbstractTopDownSet'
colData(object, ...) <- value

## S4 method for signature 'AbstractTopDownSet'
conditionData(object, ...)

## S4 replacement method for signature 'AbstractTopDownSet'
conditionData(object, ...) <- value

## S4 method for signature 'AbstractTopDownSet'
dim(x)

## S4 method for signature 'AbstractTopDownSet'
dimnames(x)

## S4 method for signature 'AbstractTopDownSet'
removeEmptyConditions(object)

## S4 method for signature 'AbstractTopDownSet'
rowViews(object, ...)
```

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

```
## S4 method for signature 'AbstractTopDownSet'
summary(object, what = c("rows", "columns"),
  ...)
```

Arguments

<code>i, j</code>	numeric, logical or character, indices specifying elements to extract or replace.
<code>drop</code>	logical, currently ignored.
<code>value</code>	replacement value.
<code>name</code>	character name of an (non)existing column in <code>colData</code> .
<code>object, x</code>	<code>AbstractTopDownSet</code>
<code>what</code>	character, specifies whether "rows" or "columns" should be summarized.
<code>...</code>	arguments passed to internal/other methods.

Details

This class just provides a common interface. It is not intended for direct use by the user. Please see [TopDownSet](#) for an example usage of its child class.

Value

This is an *Abstract/VIRTUAL* class to provide a common interface for [TopDownSet](#) and [NCBSet](#). It is not possible to create an `AbstractTopDownSet` object.

Methods (by generic)

- `[]`: Subset operator.
For `i` numeric, logical or character vectors or empty (missing) or `NULL` are supported. Subsetting is done on the fragment/bond (row) level. character indices could be names (e.g. `c("a1", "b1", "c1", "c2", "c3")`) or types (e.g. `c("c", "x")`) of the fragments for [TopDownSet](#) objects, or names of the bonds (e.g. `c("bond001")`) for [NCBSet](#) objects. `j` could be a numeric or logical vector and subsetting is done on the condition/run (column) level.
- `[[`: Subset operator.
`i` could be a numeric or logical vector and subsetting is done on the condition/run (column) level.
- `[[<-`: Setter for a column in the `colData` slot.
The `[[<-` operator is used to add/replace a single column of the `colData` `DataFrame`.
- `$`: Accessor for columns in the `colData` slot.
The `$` simplifies the accession of a single column of the `colData`. It is identical to the `[[` operator.
- `$<-`: Setter for a column in the `colData` slot.
The `$<-` operator is used to add/replace a single column of the `colData` `DataFrame`. It is identical to the `[[<-` operator.

- assayData: Accessor for the assay slot.
Returns a [Matrix::dgCMatrix](#) that stores the intensity/coverage information of [AbstractTopDownSet](#) object.
- colData: Accessor for the colData slot.
Returns a [S4Vectors::DataFrame](#) that stores metadata for the conditons/runs (columns) of the [AbstractTopDownSet](#) object.
- colData<-: Setter for the colData slot.
Replaces metadata for the conditons/runs (columns) of the [AbstractTopDownSet](#) object.
- conditionData: Accessor for the colData slot.
An alias for colData.
- conditionData<-: Setter for the colData slot.
An alias for colData<-.
- dim: Accessor for dimensions.
Returns a numeric with number of fragments/bonds (rows) and conditions/runs (columns).
- dimnames: Accessor for dimension names.
Returns a list with names for the fragments/bonds (rows) and for the conditions/runs (columns).
- removeEmptyConditions: Remove empty conditions/runs.
Removes conditions/runs (columns) without any intensity/coverage information from the [AbstractTopDownSet](#) object. It returns a modified [AbstractTopDownSet](#) object.
- rowViews: Accessor for the rowViews slot.
Depending on the implementation it returns an [FragmentViews](#) object for [TopDownSet](#) objects or an [Biostrings::XStringViews](#) object for [NCBSet](#) objects.
- summary: Summary statistics.
Returns a matrix with some statistics: number of fragments, total/min/first quartile/median/mean/third quartile/maximum of intensity values.

Slots

rowViews [Biostrings::XStringViews](#), information about fragments/bonds (name, type, sequence, mass, charge), see [Biostrings::XStringViews](#) and [FragmentViews](#) for details.

colData [S4Vectors::DataFrame](#), information about the MS2 experiments and the fragmentation conditions.

assay [Matrix::dgCMatrix](#), intensity/coverage values of the fragments/bonds. The rows correspond to the fragments/bonds and the columns to the condition/run. It just stores values that are different from zero.

files character, files that were imported.

tolerance double, tolerance in *ppm* that were used for matching the experimental *m/z* values to the theoretical fragments.

processing character, log messages.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

- [TopDownSet](#) and [NCBSet](#) which both implement/use this interface. These manual pages also provide some example code.
- [FragmentViews](#) (and [Biostrings::XStringViews](#)) for the row view interface.
- [Matrix::dgCMatrix](#) for technical details about the intensity/coverage storage.

Examples

```
## Because AbstractTopDownSet is a VIRTUAL class we could not create any
## object of it. Here we demonstrate the usage with an TopDownSet that
## implements the AbstractTopDownSet interface. See `?"TopDownSet-class"` for
## more details an further examples.

## Example data
data(tds, package="topdownr")

tds

head(summary(tds))

# Accessing slots
rowViews(tds)
colData(tds)
assayData(tds)

# Accessing colData
tds$Mz
tds$filterString

# Subsetting

# First 100 fragments
tds[1:100]

# All c fragments
tds["c"]

# Just c 152
tds["c152"]

# Condition 1 to 10
tds[, 1:10]
```

```
createTngFusionMethFiles
```

Windows specific functions.

Description

The functions `runXmlMethodChanger` and `runScanHeadsman` call `XmlMethodChanger.exe` and `ScanHeadsman.exe` with the corresponding arguments. The only work on Windows (maybe on Linux + wine as well but that was never tested).

Usage

```

createTngFusionMethFiles(template, xml = list.files(pattern = ".*\\.xml$"),
  executable = "XmlMethodChanger.exe", verbose = interactive())

runXmlMethodChanger(template, xml = list.files(pattern = ".*\\.xml$"),
  executable = "XmlMethodChanger.exe", verbose = interactive())

runScanHeadsman(path = ".", executable = "ScanHeadsman.exe")

```

Arguments

template	character, path to template .meth file.
xml	character, vector of path to .xml files.
executable	character, path to the XmlMethodChanger.exe or ScanHeadsman.exe executable.
verbose	logical, if TRUE a progress bar is shown.
path	character, path to the directory containing the .raw files.

Details

runXmlMethodChanger applies 'XmlMethodChanger.exe' on all given XML files generated with [writeMethodXmIs\(\)](#) to create .meth files from a template.

runScanHeadsman calls ScanHeadsman.exe on a given directory containing .raw files. ScanHeadsman.exe extracts the method and scan header data into .experiments.csv and .txt files, respectively.

Value

Nothing. Used for its side effects.

References

XmlMethodChanger source code: <https://github.com/thermofisherlms/meth-modifications/>

ScanHeadsman source code: <https://bitbucket.org/caetera/scanheadsman>

See Also

[writeMethodXmIs\(\)](#)

Examples

```

## Not run:
runXmlMethodChanger(templateMeth="TMS2IndependentTemplate240Extended.meth",
  modificationXml=list.files(pattern="^method.*\\.xml$"),
  executable="..\\XmlMethodChanger.exe")

## End(Not run)
## Not run:
runScanHeadsman("raw", executable="..\\ScanHeadsman.exe")

## End(Not run)

```

defaultMs1Settings *Settings for MS1/2 parameters.*

Description

This functions create the default settings for `writeMethodXmls()`.

Usage

```
defaultMs1Settings(...)
```

```
defaultMs2Settings(...)
```

Arguments

... named arguments that should be overwritten.

Value

A named list of settings.

Examples

```
# all default MS1 settings
defaultMs1Settings()

# overwrite FirstMass and set AgcTarget
defaultMs1Settings(FirstMass=100, AgcTarget=c(1e4, 1e5))

# all default MS2 settings
defaultMs2Settings()

# overwrite AgcTarget
defaultMs2Settings(AgcTarget=c(1e4, 1e5))
```

FragmentViews-class *The FragmentViews class*

Description

The FragmentViews class is a basic container for storing a set of views (start/end locations) on the same peptides/protein sequence. Additionally it keeps information about mass, type and charge of the fragments.

Usage

```
FragmentViews(sequence, mass, type, z = 1L, start = NULL, end = NULL,
              width = NULL, names = NULL, metadata = list())
```

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


Arguments

sequence	character/ Biostrings::AAString , complete protein/peptide sequence.
mass	double, mass of the fragments, same length as start/end/width.
type	character, type of the fragments, same length as start/end/width'.
z	integer, charge of the fragments, length one or same length as start/end/width'.
start	integer, start positions of the fragments. At least two of start/end/width' has to be given.
end	integer, end positions of the fragments. At least two of start/end/width' has to be given.
width	integer, width positions of the fragments. At least two of start/end/width' has to be given.
names	character, names of the fragments, same length as start/end/width'.
metadata	list, metadata like modifications.
object	FragmentViews

Details

FragmentViews extends [Biostrings::XStringViews](#). In short it combines an [IRanges::IRanges](#) object to store start/end location on a sequence, an [Biostrings::AAString](#) object.

Value

An [FragmentViews](#) object.

Functions

- [FragmentViews](#): Constructor
In general it is not necessary to call the constructor manually. See [readTopDownFiles\(\)](#) instead.

Coercion

`as(object, "data.frame")`: Coerce an [FragmentViews](#) object into a data.frame.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

[Biostrings::XStringViews](#)

Examples

```
# Constructor
fv <- FragmentViews("ACE", start=1, width=1:3, names=paste0("b", 1:3),
                    mass=c(72.04439, 232.07504, 361.11763),
                    type="b", z=1)

fv

# Coercion to data.frame
as(fv, "data.frame")
as(fv, "data.frame")
```

 NCBSet-class

The NCBSet class

Description

The NCBSet class is a container for a top-down proteomics experiment similar to the [TopDownSet](#) but instead of intensity values it just stores the information if a bond is covered by a N-terminal (encoded as 1), a C-terminal (encoded as 2) and/or bidirectional fragments (encoded as 3).

Usage

```
## S4 method for signature 'NCBSet'
bestConditions(object, n = ncol(object), minN = 0L, ...)

## S4 method for signature 'NCBSet'
fragmentationMap(object, nCombinations = 10,
  cumCoverage = TRUE, labels = colnames(object), ...)

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

## S4 method for signature 'NCBSet'
summary(object, what = c("conditions", "bonds"), ...)
```

Arguments

object	NCBSet
n	integer, max number of combinations/iterations.
minN	integer, stop if there are less than minN additional fragments
nCombinations	integer, number of combinations to show (0 to avoid plotting them at all).
cumCoverage	logical, if TRUE (default) cumulative coverage of combinations is shown.
labels	character, overwrite x-axis labels.
what	character, specifies whether "conditions" (columns; default) or "bonds" (rows) should be summarized.
...	arguments passed to internal/other methods. added. newly added fragments third column: number of newly covered bonds.

Value

An [NCBSet](#) object.

Methods (by generic)

- **bestConditions:** Best combination of conditions.
Finds the best combination of conditions for highest coverage of bonds. Use n to limit the number of iterations and combinations that should be returned. If minN is set at least minN fragments have to be added to the combinations. The function returns a 2-column matrix. The first column contains the index of the condition (column number) and the second one the newly added number of fragments.

- fragmentationMap: Plot fragmentation map.
Plots a fragmentation map of the Protein. Use nCombinations to add another plot with nCombinations combined conditions. If cumCoverage is TRUE (default) these combinations increase the coverage cumulatively.
- summary: Summary statistics.
Returns a matrix with some statistics: number of fragments, total/min/first quartile/median/mean/third quartile/maximum of intensity values.

Slots

rowViews [Biostrings::XStringViews](#), information about bonds (name, start, end, width, sequence), see [Biostrings::XStringViews](#) for details.

colData [S4Vectors::DataFrame](#), information about the MS2 experiments and the fragmentation conditions.

assay [Matrix::dgCMatrix](#), coverage values of the bonds. The rows correspond to the bonds and the columns to the condition/run. It just stores values that are different from zero. If a bond is covered by an N-terminal fragment its encoded with 1, by an C-terminal fragment with 2 and by both fragment types/bidirectional by 3 respectively.

files character, files that were imported.

tolerance double, tolerance in *ppm* that were used for matching the experimental m/z values to the theoretical fragments.

processing character, log messages.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

- An NCBSet is generated from an [TopDownSet](#) object.
- [Biostrings::XStringViews](#) for the row view interface.
- [Matrix::dgCMatrix](#) for technical details about the coverage storage.

Examples

```
## Example data
data(tds, package="topdownr")

## Aggregate technical replicates
tds <- aggregate(tds)

## Coercion into an NCBSet object
ncb <- as(tds, "NCBSet")

ncb

head(summary(ncb))

# Accessing slots
rowViews(ncb)
colData(ncb)
assayData(ncb)
```

```

# Accessing colData
ncb$Mz

# Subsetting

# First 100 bonds
ncb[1:100]

# Just bond 152
ncb["bond152"]

# Condition 1 to 10
ncb[, 1:10]

# Plot fragmentation map
fragmentationMap(ncb)

```

readTopDownFiles	<i>Read top-down files.</i>
------------------	-----------------------------

Description

It creates an [TopDownSet](#) object and is its only constructor.

Usage

```

readTopDownFiles(path, pattern = ".*", type = c("a", "b", "c", "x", "y",
"z"), modifications = c("Carbamidomethyl", "Acetyl", "Met-loss"),
adducts = data.frame(), neutralLoss = MSnbase::defaultNeutralLoss(),
sequenceOrder = c("original", "random", "inverse"), tolerance = 5e-06,
dropNonInformativeColumns = TRUE, sampleColumns = c("Mz", "AgcTarget",
"EtdReagentTarget", "EtdActivation", "CidActivation", "HcdActivation"),
verbose = interactive())

```

Arguments

path	character, path to directory that contains the top-down files.
pattern	character, a filename pattern, the default <code>.*</code> means all files.
type	character, type of fragments, currently <i>a-c</i> and <i>x-z</i> are supported, see MSnbase::calculateFragments for details.
modifications	character, unimod names of modifications that should be applied. Currently just <i>Acetyl</i> (Unimod:1 but just protein N-term), <i>Carbamidomethyl</i> (Unimod:4) and <i>Met-loss</i> (Unimod:765) are supported. <i>Met-loss</i> removes M (if followed by A, C, G, P, S, T, or V; (see also http://www.unimod.org/modifications_view.php?editid1=1 , http://www.unimod.org/modifications_view.php?editid1=4 , and http://www.unimod.org/modification for details)). Use NULL to disable all modifications.
adducts	<code>data.frame</code> , with 3 columns, namely: mass, name, to, see details section.
neutralLoss	list, neutral loss that should be applied, see MSnbase::calculateFragments() and MSnbase::defaultNeutralLoss() for details.

sequenceOrder	character, order of the sequence before fragment calculation and matching is done. "original" doesn't change anything. "inverse" reverse the sequence and "random" arranges the amino acid sequence at random.
tolerance	double, tolerance in <i>ppm</i> that is used to match the theoretical fragments with the observed ones.
dropNonInformativeColumns	logical, should columns with just one identical value across all runs be removed?
sampleColumns	character, column names of the <code>colData()</code> used to define a sample (technical replicate). This is used to add the <code>Sample</code> column (used for easier aggregation, etc.).
verbose	logical, verbose output?

Details

`readTopDownFiles` reads and processes all top-down files, namely:

- .fasta (protein sequence)
- .mzML (spectra)
- .experiments.csv (method/fragmentation conditions)
- .txt (scan header information)

adducts: *Thermo's Xtract* allows some mistakes in deisotoping, mostly it allows +/- C13-C12 and +/- H+. The `adducts` argument takes a `data.frame` with the mass to add, the name that should assign to these new fragments and an information to whom the modification should be applied, e.g. for H+ on z, `data.frame(mass=1.008, name="zpH", to="z")`.

Please note: The adducts are added to the output of `MSnbase::calculateFragments()`. That has some limitations, e.g. neutral loss calculation could not be done in [topdownr-package](#). If neutral loss should be applied on adducts you have to create additional rows, e.g.: `data.frame(mass=c(1.008, 1.008), name=c("Cterm", "D", "E", "S", "T"))`,

Value

A `TopDownSet` object.

See Also

[MSnbase::calculateFragments\(\)](#), [MSnbase::defaultNeutralLoss\(\)](#)

Examples

```
if (require("topdownrdata")) {
  # add H+ to z and no neutral loss of water
  tds <- readTopDownFiles(
    topdownrdata::topDownDataPath("myoglobin"),
    ## Use an artificial pattern to load just the fasta
    ## file and files from m/z == 1211, ETD reagent
    ## target 1e6 and first replicate to keep runtime
    ## of the example short
    pattern=".*fasta.gz$|1211_.*1e6_1",
    adducts=data.frame(mass=1.008, name="zpH", to="z"),
    neutralLoss=MSnbase::defaultNeutralLoss(
      disableWaterLoss=c("Cterm", "D", "E", "S", "T")),
    tolerance=25e-6
  )
}
```

tds

*TopDownSet Example Data***Description**

An example data set for topdownr. It is just a subset of the myoglobin dataset available in [topdownrdata::topdownrdata-package](#).

Usage

tds

Format

A [TopDownSet](#) with 2700 fragments and 351 conditions.

Details

It was created as follows:

```
tds <- readTopDownFiles(
  topdownrdata::topDownDataPath("myoglobin"),
  ## Use an artificial pattern to load just the fasta
  ## file and files from m/z == 1211, ETD reagent
  ## target 1e6 and first replicate to keep runtime
  ## of the example short
  pattern=".*fasta.gz$|1211_.*1e\+06_1",
  adducts=data.frame(mass=1.008, name="zPH", to="z"),
  neutralLoss=MSnbase::defaultNeutralLoss(
    disableWaterLoss=c("Cterm", "D", "E", "S", "T")),
  tolerance=25e-6)
```

Source

Subset taken from the [topdownrdata::topdownrdata-package](#) package.

TopDownSet-class

*The TopDownSet class***Description**

The TopDownSet class is a container for a whole top-down proteomics experiment.

Usage

```
## S4 method for signature 'TopDownSet'
aggregate(x, by = x$Sample, ...)

## S4 method for signature 'TopDownSet'
filterCv(object, threshold, by = object$Sample, ...)

## S4 method for signature 'TopDownSet'
filterInjectionTime(object, maxDeviation = log2(3),
  keepTopN = 2, by = object$Sample, ...)

## S4 method for signature 'TopDownSet'
filterIntensity(object, threshold, relative = TRUE,
  ...)

## S4 method for signature 'TopDownSet'
filterNonReplicatedFragments(object, minN = 2,
  by = object$Sample, ...)

## S4 method for signature 'TopDownSet'
normalize(object, method = "TIC", ...)

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

## S4 method for signature 'TopDownSet'
summary(object, what = c("conditions", "fragments"),
  ...)
```

Arguments

x, object	TopDownSet
by	list, grouping variable, in general it refers to technical
threshold	double, threshold variable.
maxDeviation	double, maximal allowed deviation in the log2 injection time in ms in comparison to the median ion injection time.
keepTopN	integer, how many technical replicates should be kept?
relative	logical, if relative is TRUE all fragments with intensity below threshold * max(intensity) per fragment are removed, otherwise all fragments below threshold are removed.
minN	numeric, if less than minN of a fragment are found across technical replicates it is removed.
method	character, normalisation method, currently just "TIC" for Total Ion Current normalisation of the scans/conditions (column-wise normalisation) is supported.
what	character, specifies whether "conditions" (columns; default) or "fragments" (rows) should be summarized.
...	arguments passed to internal/other methods. replicates (that's why the default is the "Sample" column in colData).

Details

See `vignette("analysis", package="topdownr")` for a detailed example how to work with TopDownSet objects.

Value

An [TopDownSet](#) object.

Methods (by generic)

- `aggregate`: Aggregate conditions/runs.
Aggregates conditions/runs (columns) in an [TopDownSet](#) object by a user-given value (default is the "Sample" column of `colData` which has the same value for technical replicates). It combines intensity values and numeric metadata of the grouped conditions/runs (columns) by mean and returns a reduced [TopDownSet](#) object.
- `filterCv`: Filter by CV.
Filtering is done by coefficient of variation across technical replicates (defined by the `by` argument). All fragments below a given threshold are removed. The threshold is the maximal allowed CV in percent ($sd/mean * 100$ < threshold).
- `filterInjectionTime`: Filter by ion injection time.
Filtering is done by maximal allowed deviation and just the technical keepTopN replicates with the lowest deviation from the median ion injection time are kept.
- `filterIntensity`: Filter by intensity.
Filtering is done by removing all fragments that are below a given (absolute/relative) intensity threshold.
- `filterNonReplicatedFragments`: Filter by non-replicated fragments.
Filtering is done by removing all fragments that don't replicate across technical replicates.
- `normalize`: Normalise.
Applies *Total Ion Current* normalisation to a [TopDownSet](#) object. The normalisation is done per scans/conditions (column-wise normalisation).
- `summary`: Summary statistics.
Returns a matrix with some statistics: number of fragments, total/min/first quartile/median/mean/third quartile/maximum of intensity values.

Slots

`rowViews` [FragmentViews](#), information about fragments (name, type, sequence, mass, charge), see [FragmentViews](#) for details.

`colData` [S4Vectors::DataFrame](#), information about the MS2 experiments and the fragmentation conditions.

`assay` [Matrix::dgCMatrix](#), intensity values of the fragments. The rows correspond to the fragments and the columns to the condition/run. It just stores values that are different from zero.

`files` character, files that were imported.

`tolerance` double, tolerance in *ppm* that were used for matching the experimental *mz* values to the theoretical fragments.

`processing` character, log messages.

Coercion

`'as(object, "MSnSet")`: Coerce an [TopDownSet](#) object into an [MSnbase::MSnSet](#) object.

`'as(object, "NCBSet")`: Coerce an [TopDownSet](#) object into an [NCBSet](#) object.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>

See Also

- [FragmentViews](#) for the row view interface.
- [Matrix::dgCMatrix](#) for technical details about the intensity storage.
- `?vignette("analysis", package="topdownr")` for a full documented example of an analysis using topdownr.

Examples

```
## Example data
data(tds, package="topdownr")

tds

head(summary(tds))

# Accessing slots
rowViews(tds)
colData(tds)
assayData(tds)

# Accessing colData
tds$Mz
tds$filterString

# Subsetting

# First 100 fragments
tds[1:100]

# All c fragments
tds["c"]

# Just c 152
tds["c152"]

# Condition 1 to 10
tds[, 1:10]

# Filtering
# Filter all intensities that don't have at least 10 % of the highest
# intensity per fragment.
tds <- filterIntensity(tds, threshold=0.1)

# Filter all conditions with a CV above 30 % (across technical replicates)
tds <- filterCv(tds, threshold=30)
```

```

# Filter all conditions with a large deviation in injection time
tds <- filterInjectionTime(tds, maxDeviation=log2(3), keepTopN=2)

# Filter all conditions where fragments don't replicate
tds <- filterNonReplicatedFragments(tds)

# Normalise by TIC
tds <- normalize(tds)

# Aggregate technical replicates
tds <- aggregate(tds)

head(summary(tds))

# Coercion
as(tds, "NCBSet")

if (require("MSnbase")) {
  as(tds, "MSnSet")
}

```

writeMethodXmIs

Create Orbitrap Fusion method.xml files.

Description

This function is used to create Orbitrap Fusion method files from all combinations of a user-given set of MS1 and MS2 settings.

Usage

```

writeMethodXmIs(ms1Settings, ms2Settings, groupBy = c("replication",
  "ETDReactionTime"), replications = 2, mz, massLabeling = TRUE,
  nMs2perMs1 = 10, duration = 0.5, randomise = TRUE,
  pattern = "method_%s.xml", verbose = interactive())

```

Arguments

ms1Settings	list, named list of MS1 settings, see e.g. defaultMs1Settings() .
ms2Settings	list, named list of MS2 settings, see e.g. defaultMs2Settings() .
groupBy	character, split files by groupBy entries.
replications	integer, number of replications of experiments.
mz	matrix, two columns (column 1: <i>mass</i> , column 2: <i>z</i>).
massLabeling	logical, should <i>mz</i> values used for ID labeling?
nMs2perMs1	integer, how many MS2 scans should be run after a MS1 scan?
duration	double, how long should the scan be?
randomise	logical, should the MS2 scan settings randomised?
pattern	character, file name pattern for the method.xml files.
verbose	logical, verbose output?

Details

- `ms1Settings`: A list of MS1 settings. This has to be a named list. Valid MS1 settings are: `c("FirstMass", "LastMass", "Microscans", "MaxITTimeInMS", "AgcTarget")`
- `ms2Settings`: A list of MS2 settings. This has to be a named list. Valid MS2 settings are: `c("ActivationType", "IsolationWindow", "EnableMultiplexIons", "EnableMSXIds", "MaxNoOfMultiplexIons")`
- `groupBy`: The `groupBy` parameter is used to split methods into different files. Valid entries are all settings that could be used in `ms2Settings` and "replication".
- `massLabeling`: The Orbitrap Fusion devices seems not to respect the start and end times of the runs given in the `method.xml` files. That's why it is nearly impossible to identify the run with its conditions based on the timings. If `massLabeling` is TRUE (default) the mass values given in `mz` are rounded to the first decimal place and the second to fourth decimal place is used as numeric identifier.
- `pattern`: The file name pattern used to name different method files. It must contain a "%s" that is replaced by the conditions defined in `groupBy`.

Value

An invisible list with the MS1, MS2, runtimes and `mz`.

Author(s)

Sebastian Gibb <mail@sebastiangibb.de>, Pavel V. Shliaha <pavels@bmb.sdu.dk>

See Also

[defaultMs1Settings\(\)](#), [defaultMs2Settings\(\)](#)

Examples

```
writeMethodXmIs(defaultMs1Settings(FirstMass=400),
                defaultMs2Settings(),
                mz=cbind(mass=c(609.21, 700.45, 823.95), z=10),
                groupBy="ETDReactionTime",
                replications=1,
                pattern="method_firstmass_400_%s.xml")
```

Index

- *Topic **datasets**
 - tds, [14](#)
- *Topic **package**
 - topdownr-package, [2](#)
- [, AbstractTopDownSet, ANY, ANY, ANY-method (AbstractTopDownSet-class), [3](#)
- [[, AbstractTopDownSet, ANY, missing, -method (AbstractTopDownSet-class), [3](#)
- [[, AbstractTopDownSet, ANY, missing-method (AbstractTopDownSet-class), [3](#)
- [[<-, AbstractTopDownSet, ANY, missing, -method (AbstractTopDownSet-class), [3](#)
- [[<-, AbstractTopDownSet, ANY, missing-method (AbstractTopDownSet-class), [3](#)
- \$, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- \$<-, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)

- AbstractTopDownSet, [5](#)
- AbstractTopDownSet-class, [3](#)
- aggregate, TopDownSet-method (TopDownSet-class), [14](#)
- assayData, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)

- bestConditions (NCBSet-class), [10](#)
- bestConditions, NCBSet-method (NCBSet-class), [10](#)
- Biostrings::AAString, [9](#)
- Biostrings::XStringViews, [5, 6, 9, 11](#)

- coerce, FragmentViews, data.frame-method (FragmentViews-class), [8](#)
- coerce, TopDownSet, MSnSet-method (TopDownSet-class), [14](#)
- coerce, TopDownSet, NCBSet-method (TopDownSet-class), [14](#)
- colData (AbstractTopDownSet-class), [3](#)
- colData(), [13](#)
- colData, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- colData<- (AbstractTopDownSet-class), [3](#)

- colData<-, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- conditionData (AbstractTopDownSet-class), [3](#)
- conditionData, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- conditionData<- (AbstractTopDownSet-class), [3](#)
- conditionData<-, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- createTngFusionMethFiles, [6](#)

- defaultMs1Settings, [8](#)
- defaultMs1Settings(), [18, 19](#)
- defaultMs2Settings (defaultMs1Settings), [8](#)
- defaultMs2Settings(), [18, 19](#)
- dim, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)
- dimnames, AbstractTopDownSet-method (AbstractTopDownSet-class), [3](#)

- filterCv (TopDownSet-class), [14](#)
- filterCv, TopDownSet-method (TopDownSet-class), [14](#)
- filterInjectionTime (TopDownSet-class), [14](#)
- filterInjectionTime, TopDownSet-method (TopDownSet-class), [14](#)
- filterIntensity (TopDownSet-class), [14](#)
- filterIntensity, TopDownSet-method (TopDownSet-class), [14](#)
- filterNonReplicatedFragments (TopDownSet-class), [14](#)
- filterNonReplicatedFragments, TopDownSet-method (TopDownSet-class), [14](#)
- fragmentationMap (NCBSet-class), [10](#)
- fragmentationMap, NCBSet-method (NCBSet-class), [10](#)
- FragmentViews, [5, 6, 9, 16, 17](#)
- FragmentViews (FragmentViews-class), [8](#)
- FragmentViews-class, [8](#)

- IRanges::IRanges, [9](#)

Matrix::dgCMatrix, [5](#), [6](#), [11](#), [16](#), [17](#)
MSnbase::calculateFragments(), [12](#), [13](#)
MSnbase::defaultNeutralLoss(), [12](#), [13](#)
MSnbase::MSnSet, [17](#)

NCBSet, [3–6](#), [10](#), [17](#)
NCBSet-class, [10](#)
normalize, TopDownSet-method
 (TopDownSet-class), [14](#)

readTopDownFiles, [12](#)
readTopDownFiles(), [9](#)
removeEmptyConditions
 (AbstractTopDownSet-class), [3](#)
removeEmptyConditions, AbstractTopDownSet-method
 (AbstractTopDownSet-class), [3](#)
rowViews (AbstractTopDownSet-class), [3](#)
rowViews, AbstractTopDownSet-method
 (AbstractTopDownSet-class), [3](#)
runScanHeadsman
 (createTngFusionMethFiles), [6](#)
runXmlMethodChanger
 (createTngFusionMethFiles), [6](#)

S4Vectors::DataFrame, [5](#), [11](#), [16](#)
show, AbstractTopDownSet-method
 (AbstractTopDownSet-class), [3](#)
show, FragmentViews-method
 (FragmentViews-class), [8](#)
show, NCBSet-method (NCBSet-class), [10](#)
show, TopDownSet-method
 (TopDownSet-class), [14](#)
summary, AbstractTopDownSet-method
 (AbstractTopDownSet-class), [3](#)
summary, NCBSet-method (NCBSet-class), [10](#)
summary, TopDownSet-method
 (TopDownSet-class), [14](#)

tds, [14](#)
topdownr (topdownr-package), [2](#)
topdownr-package, [2](#), [13](#)
topdownrdata::topdownrdata-package, [14](#)
TopDownSet, [3–6](#), [10–12](#), [14](#), [16](#), [17](#)
TopDownSet-class, [14](#)

writeMethodXmIs, [18](#)
writeMethodXmIs(), [7](#), [8](#)