

# Package ‘RMassBank’

January 24, 2026

**Type** Package

**Title** Workflow to process tandem MS files and build MassBank records

**Version** 3.21.0

**Author** Michael Stravs, Emma Schymanski, Steffen Neumann, Erik Mueller, Paul Stahlhofen, Tobias Schulze with contributions of Hendrik Treutler

**Maintainer** RMassBank at Eawag <massbank@eawag.ch>

**Description** Workflow to process tandem MS files and build MassBank records. Functions include automated extraction of tandem MS spectra, formula assignment to tandem MS fragments, recalibration of tandem MS spectra with assigned fragments, spectrum cleanup, automated retrieval of compound information from Internet databases, and export to MassBank records.

**License** Artistic-2.0

**Depends** Rcpp

**Imports** assertthat, Biobase, ChemmineR, data.table, digest, dplyr, enviPat, glue, httr, httr2, logger, methods, MSnbase, mzR, purrr, R.utils, rcdk, readJDX, readr, rjson, S4Vectors, tibble, tidyselect, webchem, XML, yaml

**Suggests** BiocStyle, CAMERA, gplots, knitr, magick, rmarkdown, RMassBankData (>= 1.33.1), RUnit, xcms (>= 1.37.1)

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, Bioinformatics, MassSpectrometry, Metabolomics, Software

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**SystemRequirements** OpenBabel

**Collate** 'alternateAnalyze.R' 'formulaCalculator.R' 'getSplash.R'  
'leCsvAccess.R' 'settings\_example.R' 'webAccess.R'  
'deprofile.R' 'parseMassBank.R' 'Generics.R'  
'SpectrumClasses.R' 'SpectrumMethods.R' 'RmbSpectrum2Update.R'  
'readWriteMgfData.R' 'RmbWorkspace.R' 'RmbWorkspaceUpdate.R'

```
'SpectraSetMethods.R' 'AggregateMethods.R' 'leMsMs.r'
'leMsmsRaw.R' 'buildRecord.R' 'createMassBank.R'
'msmsRawExtensions.r' 'validateMassBank.R' 'tools.R'
'msmsRead.R' 'mergeSpectra.R' 'Isotopic_Annotation.R'
'fillback.R' 'parseMbRecord.R' 'zzz.R' 'log_wrapper.R'
'createCompoundlist.R'
```

**git\_url** <https://git.bioconductor.org/packages/RMassBank>

**git\_branch** devel

**git\_last\_commit** c946a1f

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2026-01-23

## Contents

+,RmbSpectraSet,ANY-method . . . . .	5
+,RmbSpectrum2List,ANY-method . . . . .	5
+,Spectrum,numeric-method . . . . .	6
-,RmbSpectraSet,ANY-method . . . . .	6
-,RmbSpectrum2List,ANY-method . . . . .	7
-,Spectrum,numeric-method . . . . .	7
.parseTitleString . . . . .	8
.updateObject.RmbSpectrum2.formulaSource . . . . .	9
add.formula . . . . .	9
addMB . . . . .	10
addPeaks . . . . .	11
addPeaksManually . . . . .	12
addProperty . . . . .	13
aggregateSpectra . . . . .	14
analyzeMsMs . . . . .	15
annotator.default . . . . .	18
archiveResults . . . . .	19
buildRecord . . . . .	19
CAS2SMILES . . . . .	21
checkIsotopes . . . . .	21
checkSpectra . . . . .	23
cleanElnoise . . . . .	24
combineMultiplicities . . . . .	25
compoundlist2SDF . . . . .	26
createCompoundlist . . . . .	27
createMolfile . . . . .	28
CTS.externalIdSubset . . . . .	29
CTS.externalIdTypes . . . . .	30
dbe . . . . .	30
deprofile . . . . .	31
exportMassbank . . . . .	33

fillback	34
filterCompoundlist	35
filterLowaccResults	36
filterMultiplicity	37
filterPeakSatellites	38
filterPeaksMultiplicity	39
findEIC	40
findMass	42
findMsMsHR	43
findMsMsHR.direct	46
findMsMsHR.ticms2	47
findMsMsHRperMsp	48
findMsMsHRperxcms	49
findMz	50
findMz.formula	52
findProgress	53
flatten	53
formulastring.to.list	55
gatherCCTE	56
gatherData	57
gatherDataBabel	58
gatherDataUnknown	59
gatherPubChem	60
getAnalyticalInfo	61
getCactus	61
getCASRN	62
getCSID	63
getCtsKey	64
getCtsRecord	65
getData	66
getDTXCID	66
getDTXSID	67
getDTXSMILES	68
getField	69
getMolecule	70
getPcId	71
getPrefName	72
is.valid.formula	73
loadInfolists	74
loadList	75
makeMollist	76
makePeaksCache	76
makeRecalibration	77
mbWorkflow	79
mbWorkspace-class	80
mergePeaks	81
mergeSpectra	82
msmsRead	83

msmsRead.RAW	85
msmsWorkflow	86
msmsWorkspace-class	88
newMbWorkspace	89
newMsmsWorkspace	89
normalize,RmbSpectrum2-method	90
normalize,RmbSpectrum2List-method	91
order.formula	91
parseMassBank	92
parseMbRecord	93
peaksMatched	94
peaksUnmatched	94
plotMbWorkspaces	95
plotRecalibration	96
ppm	97
problematicPeaks	98
processProblematicPeaks	99
progressBarHook	99
property	100
property<-	101
reanalyzeFailpeaks	102
recalibrate	103
recalibrate.addMS1data	105
RmbDefaultSettings	106
RmbSettings	107
RmbSpectraSet-class	109
RmbSpectrum2-class	110
RmbSpectrum2List-class	111
rmb_log_debug	111
rmb_log_error	112
rmb_log_fatal	112
rmb_log_info	113
rmb_log_success	113
rmb_log_trace	114
rmb_log_warn	114
selectPeaks	115
selectSpectra	116
setAccessionBuilder	117
setData	117
smiles2mass	118
spectraCount	119
to.limits.rcdk	120
toMassbank	121
toRMB	122
updateHeader	123
updateSettings	124
validate	125

---

`+,RmbSpectraSet,ANY-method`

*Add a mass shift to a list of spectra*

---

## Description

Shifts both ‘parent’ and ‘children’ spectra of the ‘RmbSpectraSet’ by the same mass.

## Usage

```
## S4 method for signature 'RmbSpectraSet,ANY'  
e1 + e2
```

## Arguments

e1	a ‘RmbSpectraSet’ object containing zero or more ‘children’ spectra and a ‘parent’ spectrum
e2	a numeric mass shift

---

`+,RmbSpectrum2List,ANY-method`

*Add a mass shift to a list of spectra*

---

## Description

Shifts all spectra in a ‘RmbSpectrum2List’ by the same mass

## Usage

```
## S4 method for signature 'RmbSpectrum2List,ANY'  
e1 + e2
```

## Arguments

e1	a ‘RmbSpectrum2List’ object containing zero or more ‘RmbSpectrum2’ spectra
e2	a numeric mass shift

---

**+,Spectrum,numeric-method**  
*Add a mass shift to a spectrum*

---

### Description

Add a mass shift to a spectrum

### Usage

```
## S4 method for signature 'Spectrum,numeric'
e1 + e2
```

### Arguments

e1	a 'MSnbase::Spectrum' object
e2	a numeric mass shift

---

**-,RmbSpectraSet,ANY-method**  
*Add a negative mass shift to a list of spectra*

---

### Description

Shifts both 'parent' and 'children' spectra of the 'RmbSpectraSet' by the same mass.

### Usage

```
## S4 method for signature 'RmbSpectraSet,ANY'
e1 - e2
```

### Arguments

e1	a 'RmbSpectraSet' object containing zero or more 'children' spectra and a 'parent' spectrum
e2	a numeric mass shift

---

-,RmbSpectrum2List,ANY-method

*Add a negative mass shift to a list of spectra*

---

## Description

Shifts all spectra in a ‘RmbSpectrum2List‘ by the same mass

## Usage

```
## S4 method for signature 'RmbSpectrum2List,ANY'  
e1 - e2
```

## Arguments

e1	a ‘RmbSpectrum2List‘ object containing zero or more ‘RmbSpectrum2‘ spectra
e2	a numeric mass shift

---

-,Spectrum,numeric-method

*Add a negative mass shift to a spectrum*

---

## Description

Add a negative mass shift to a spectrum

## Usage

```
## S4 method for signature 'Spectrum,numeric'  
e1 - e2
```

## Arguments

e1	a ‘MSnbase::Spectrum‘ object
e2	a numeric mass shift

---

<code>.parseTitleString</code>	<i>Parse record title</i>
--------------------------------	---------------------------

---

## Description

Parses a title for a single MassBank record using the title format specified in the option `titleFormat`. Internally used, not exported.

## Usage

```
.parseTitleString(mbdata)
```

## Arguments

<code>mbdata</code>	list The information data block for the record header, as stored in <code>mbdata_relisted</code> after loading an infolist.
---------------------	---

## Details

If the option is not set, a standard title format is used (for record definition version 1 or 2).

## Value

A string with the title.

## Author(s)

Michael Stravs, Eawag

## References

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

## See Also

[buildRecord](#)

## Examples

```
## Not run:  
# used in buildRecord()  
title <- .parseTitleString(mbdata)  
  
## End(Not run)
```

---

.updateObject.RmbSpectrum2.formulaSource  
*Add formulaSource column to spectrum.*

---

## Description

TODO: consider whether to add functionality to move reanalysis stuff from legacy data back in.

## Usage

.updateObject.RmbSpectrum2.formulaSource(w)

## Arguments

w RmbSpectrum2 The object to be updated

## Value

The updated RmbSpectrum2 object

## Author(s)

stravsmi

---

add.formula *Calculations on molecular formulas*

---

## Description

Add, subtract, and multiply molecular formulas.

## Usage

add.formula(f1, f2, as.formula = TRUE, as.list = FALSE)  
multiply.formula(f1, n, as.formula = TRUE, as.list = FALSE)

## Arguments

f1, f2 Molecular formulas (in list form or in text form) to calculate with.  
as.formula Return the result as a text formula (e.g. "C6H12O6"). This is the default  
as.list Return the result in list format (e.g. list(C=6, H=12, O=6)).  
n Multiplier (positive or negative, integer or non-integer.)

## Details

Note that the results are not checked for plausibility at any stage, nor reordered.

**Value**

The resulting formula, as specified above.

**Author(s)**

Michael Stravs

**See Also**

[formula](#)[string](#)[.to.list](#), [is.valid.formula](#), [order.formula](#)

**Examples**

```
##  
  
add.formula("C6H12O6", "C3H3")  
add.formula("C6H12O6", "C-3H-3")  
add.formula("C6H12O6", multiply.formula("C3H3", -1))
```

---

**addMB***MassBank-record Addition*

---

**Description**

Adds the peaklist of a MassBank-Record to the specs of an msmsWorkspace

**Usage**

addMB(w, cpdID, fileName, mode)

**Arguments**

w	The msmsWorkspace that the peaklist should be added to.
cpdID	The compoundID of the compound that has been used for the record
fileName	The path to the record
mode	The ionization mode that has been used to create the record

**Value**

The msmsWorkspace with the additional peaklist from the record

**Author(s)**

Erik Mueller

**See Also**[addPeaksManually](#)**Examples**

```
## Not run:  
addMB("filepath_to_records/RC00001.txt")  
  
## End(Not run)
```

---

**addPeaks***Add additional peaks to spectra*

---

**Description**

Loads a table with additional peaks to add to the MassBank spectra. Required columns are cpdID, scan, int, mzFound, OK.

**Usage**

```
addPeaks(mb, filename_or_dataframe)
```

**Arguments**

<code>mb</code>	The mbWorkspace to load the peaks into.
<code>filename_or_dataframe</code>	Filename of the csv file, or name of the R dataframe containing the peaklist.

**Details**

All peaks with OK=1 will be included in the spectra.

**Value**

The mbWorkspace with loaded additional peaks.

**Author(s)**

Michael Stravs

**See Also**[mbWorkflow](#)**Examples**

```
## Not run: addPeaks("myrun_additionalPeaks.csv")
```

---

**addPeaksManually** *Addition of manual peaklists*

---

**Description**

Adds a manual peaklist in matrix-format

**Usage**

```
addPeaksManually(w, cpdID, handSpec, mode)
```

**Arguments**

w	The msmsWorkspace that the peaklist should be added to.
cpdID	The compoundID of the compound that has been used for the peaklist
handSpec	A peaklist with 2 columns, one with "mz", one with "int"
mode	The ionization mode that has been used for the spectrum represented by the peaklist

**Value**

The msmsWorkspace with the additional peaklist added to the right spectrum

**Author(s)**

Erik Mueller

**See Also**

[msmsWorkflow](#)

**Examples**

```
## Not run:  
handSpec <- cbind(mz=c(274.986685367956, 259.012401087427, 95.9493025990907, 96.9573002472772),  
                    int=c(357,761, 2821, 3446))  
addPeaksManually(w, cpdID, handSpec)  
  
## End(Not run)
```

---

addProperty	<i>Add and initialize dataframe column</i>
-------------	--

---

## Description

Adds a new column of a defined type to a `data.frame` and initializes it to a value. The advantage of doing this over adding it with `$` or `[, ""]` is that the case `nrow(o) == 0` is adequately handled and doesn't raise an error.

## Usage

```
addProperty(o, name, type, value = NA)

## S4 method for signature 'RmbSpectrum2,character,character'
addProperty(o, name, type, value = NA)

## S4 method for signature 'data.frame,character,character'
addProperty(o, name, type, value = NA)
```

## Arguments

<code>o</code>	<code>data.frame</code> to add the column to
<code>name</code>	Name of the new column
<code>type</code>	Data type of the new column
<code>value</code>	Initial value of the new column (NA if not given)

## Value

Expanded data frame.

## Methods (by class)

- `addProperty(o = data.frame, name = character, type = character)`: Add a new column to a `data.frame`

## Author(s)

stravsmi

---

aggregateSpectra	<i>Aggregate analyzed spectra</i>
------------------	-----------------------------------

---

## Description

Groups an array of analyzed spectra and creates aggregated peak tables

## Usage

```
aggregateSpectra(spec, addIncomplete=FALSE)
```

## Arguments

spec	The RmbSpectraSetList of spectra sets (RmbSpectraSet objects) to aggregate
addIncomplete	Whether or not the peaks from incomplete files (files for which less than the maximal number of spectra are present)

## Details

*addIncomplete* is relevant for recalibration. For recalibration, we want to use only high-confidence peaks, therefore we set *addIncomplete* to FALSE. When we want to generate a peak list for actually generating MassBank records, we want to include all peaks into the peak tables.

## Value

A summary data.frame with all peaks (possibly multiple rows for one m/z value from a spectrum, see below) with columns:

mzFound, intensity	Mass and intensity of the peak
good	if the peak passes filter criteria
mzCalc, formula, dbe, dppm	calculated mass, formula, dbe and ppm deviation of the assigned formula
formulaCount, dppmBest	Number of matched formulae for this m/z value, and ppm deviation of the best match
scan, cpdID, parentScan	Scan number of the child and parent spectrum in the raw file, also the compound ID to which the peak belongs
dppmRc	ppm deviation recalculated from the aggregation function
index	Aggregate-table peak index, so the table can be subsetted, edited and results reinserted back into this table easily

Further columns are later added by workflow steps 6 (electronic noise culler), 7 and 8.

**Author(s)**

Michael Stravs

**See Also**

[msmsWorkflow](#), [analyzeMsMs](#)

**Examples**

```
## As used in the workflow:  
## Not run: %  
w@spectra <- lapply(w@spectra, function(spec)  
  analyzeMsMs(spec, mode="pH", detail=TRUE, run="recalibrated", cut=0, cut_ratio=0 ) )  
w@aggregate <- aggregateSpectra(w@spectra)  
  
## End(Not run)
```

---

analyzeMsMs

*Analyze MSMS spectra*

---

**Description**

Analyzes MSMS spectra of a compound by fitting formulas to each subpeak.

**Usage**

```
analyzeMsMs(  
  msmsPeaks,  
  mode = "pH",  
  detail = FALSE,  
  run = "preliminary",  
  filterSettings = getOption("RMassBank")$filterSettings,  
  spectralList = getOption("RMassBank")$spectralList,  
  method = "formula"  
)  
  
analyzeMsMs.formula(  
  msmsPeaks,  
  mode = "pH",  
  detail = FALSE,  
  run = "preliminary",  
  filterSettings = getOption("RMassBank")$filterSettings  
)  
  
analyzeMsMs.intensity(  
  msmsPeaks,
```

```

  mode = "pH",
  detail = FALSE,
  run = "preliminary",
  filterSettings = getOption("RMassBank")$filterSettings
)

```

## Arguments

msmsPeaks	A RmbSpectraSet object. Corresponds to a parent spectrum and children MSMS spectra of one compound (plus some metadata). The objects are typically generated with <a href="#">findMsMsHR</a> , and populate the @spectrum slot in a msmsWorkspace (refer to the corresponding documentation for the precise format specifications).
mode	Specifies the processing mode, i.e. which molecule species the spectra contain. <i>pH</i> (positive H) specifies $[M+H]^+$ , <i>pNa</i> specifies $[M+Na]^+$ , <i>pM</i> specifies $[M]^+$ , <i>mH</i> and <i>mNa</i> specify $[M-H]^-$ and $[M-Na]^-$ , respectively. (I apologize for the naming of <i>pH</i> which has absolutely nothing to do with chemical <i>pH</i> values.)
detail	Whether detailed return information should be provided (defaults to FALSE). See below.
run	"preliminary" or "recalibrated". In the preliminary run, mass tolerance is set to 10 ppm (above m/z 120) and 15 ppm (below m/z 120), the default intensity cutoff is $\$10^{4\$}$ for positive mode (no default cutoff in negative mode), and the column "mz" from the spectra is used as data source. In the recalibrated run, the mass tolerance is set to 5 ppm over the whole mass range, the default cutoff is 0 and the column "mzRecal" is used as source for the m/z values. Defaults to "preliminary".
filterSettings	Settings for the filter parameters, by default loaded from the RMassBank settings set with e.g. <a href="#">loadRmbSettings</a> . Must contain: <ul style="list-style-type: none"> <li>• ppmHighMass, allowed ppm deviation before recalibration for high mass range</li> <li>• ppmLowMass, allowed ppm deviation before recalibration for low mass range</li> <li>• massRangeDivision, division point between high and low mass range (before recalibration)</li> <li>• ppmFine, allowed ppm deviation overall after recalibration</li> <li>• prelimCut, intensity cutoff for peaks in preliminary run</li> <li>• prelimCutRatio, relative intensity cutoff for peaks in preliminary run, e.g. 0.01 = 1</li> <li>• fineCut, intensity cutoff for peaks in second run</li> <li>• fineCutRatio, relative intensity cutoff for peaks in second run</li> <li>• specOkLimit, minimum intensity of base peak for spectrum to be accepted for processing</li> <li>• dbeMinLimit, minimum double bond equivalent for accepted molecular subformula.</li> <li>• satelliteMzLimit, for satellite peak filtering (<a href="#">filterPeakSatellites</a>: mass window to use for satellite removal)</li> <li>• satelliteIntLimit, the relative intensity below which to discard "satellites". (refer to <a href="#">filterPeakSatellites</a>).</li> </ul>

spectraList	The list of MS/MS spectra present in each data block. As also defined in the settings file.
method	Selects which function to actually use for data evaluation. The default "formula" runs a full analysis via formula assignment to fragment peaks. The alternative setting "intensity" calls a "mock" implementation which circumvents formula assignment and filters peaks purely based on intensity cutoffs and the satellite filtering. (In this case, the ppm and dbe related settings in filterSettings are ignored.)

### Details

The analysis function uses Rcdk. Note that in this step, *satellite peaks* are removed by a simple heuristic rule (refer to the documentation of [filterPeakSatellites](#) for details.)

### Value

The processed RmbSpectraSet object. Added (or filled, respectively, since the slots are present before) data include

```
list("complete")
  whether all spectra have useful value
list("empty")  whether there are no useful spectra
list("children")
  The processed RmbSpectrum2 objects (in a RmbSpectrum2List).
  • ok if the spectrum was successfully processed with at least one resulting peak
  • mz, intensity: note that mz/int pairs can be duplicated when multiple matches are found for one mz value, therefore the two slots are not necessarily unchanged from before
  • rawOK (logical) whether the m/z peak passes satellite/low removal
  • low, satellite if TRUE, the peak failed cutoff (low) or was removed as satellite
  • formula, mzCalc, dppm, dbe Formula, calculated mass, ppm deviation and dbe assigned to a peak
  • formulaCount, dppmBest Number of formulae matched for this m/z value and ppm deviation of the best match
  • info Spectrum identifying information (collision energy, resolution, collision mode) from the spectraList
  • All other entries are retained from the original RmbSpectrum2.
```

### Functions

- analyzeMsMs.formula(): Analyze the peaks using formula annotation
- analyzeMsMs.intensity(): Analyze the peaks going only by intensity values

### Author(s)

Michael Stravs

**See Also**

[msmsWorkflow](#), [filterLowaccResults](#), [filterPeakSatellites](#), [reanalyzeFailpeaks](#)

**Examples**

```
## Not run: analyzed <- analyzeMsMs(spec, "pH", TRUE)
```

---

annotator.default

*Generate peak annotation from peaklist*

---

**Description**

Generates the PK\$ANNOTATION entry from the peaklist obtained. This function is overridable by using the "annotator" option in the settings file.

**Usage**

```
annotator.default(annotation, formulaTag)
```

**Arguments**

annotation	A peak list to be annotated. Contains columns: "cpdID", "formula", "mzFound", "scan", "mzCalc", "dppm", "dbe", "mz", "int", "formulaCount", "parentScan", "fM_factor", "formulaMultiplicity", "intrel", "mzSpec"
formulaTag	The ion type to be added to annotated formulas ("+" or "-" usually)

**Value**

The annotated peak table. Table `colnames()` will be used for the titles (preferably don't use spaces in the column titles; however no format is strictly enforced by the MassBank data format).

**Author(s)**

Michele Stravs, Eawag <stravsmi@eawag.ch>

**Examples**

```
## Not run:
annotation <- annotator.default(annotation)

## End(Not run)
```

---

archiveResults	<i>Backup msmsWorkflow results</i>
----------------	------------------------------------

---

### Description

Writes the results from different msmsWorkflow steps to a file.

### Usage

```
archiveResults(w, fileName, settings = getOption("RMassBank"))
```

### Arguments

w	The msmsWorkspace to be saved.
fileName	The filename to store the results under.
settings	The settings to be stored into the msmsWorkspace image.

### Examples

```
# This doesn't really make a lot of sense,  
# it stores an empty workspace.  
RmbDefaultSettings()  
w <- newMsmsWorkspace()  
archiveResults(w, "narcotics.RData")
```

---

buildRecord	<i>Build MassBank records</i>
-------------	-------------------------------

---

### Description

Takes a spectra block for a compound, as returned from [analyzeMsMs](#), and an aggregated cleaned peak table, together with a MassBank information block, as stored in the infolists and loaded via [loadInfolist/readMbdata](#) and processes them to a MassBank record

### Usage

```
buildRecord(o, ..., cpd, mbdata, analyticalInfo, additionalPeaks)  
  
## S4 method for signature 'RmbSpectraSet'  
buildRecord(o, ..., cpd, mbdata, analyticalInfo, additionalPeaks)  
  
## S4 method for signature 'RmbSpectrum2'  
buildRecord(  
  o,  
  ...,
```

```

cpd = NULL,
mbdata = list(),
analyticalInfo = list(),
additionalPeaks = NULL
)

```

## Arguments

- o RmbSpectraSet or RmbSpectrum2 The spectra (or single spectrum) should be taken from a compound after analysis ([analyzeMsMs](#)). Note that **peaks are not read from this object anymore**: Peaks come from the aggregated dataframe (and from the global additionalPeaks dataframe; cf. [addPeaks](#) for usage information.)
- ... keyword arguments for intensity normalization and peak selection (see [normalize](#) and [selectPeaks](#))
- cpd RmbSpectraSet or missing In case o is an RmbSpectrum2, this represents the RmbSpectraSet it belongs to
- mbdata list The information data block for the record header, as stored in mbdata\_relisted after loading an infolist.
- analyticalInfo A list containing information for the 'AC\$' section of a MassBank record, with elements 'ai, ac\_lc, ac\_ms' for general, LC and MS info respectively.
- additionalPeaks data.frame If present, a table with additional peaks to add into the spectra. As loaded with [addPeaks](#).

## Value

An object of the same type as was used for the input with new information added to it

## Author(s)

Michael Stravs

## References

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

## See Also

[mbWorkflow](#), [addPeaks](#), [toMassbank](#)

---

**CAS2SMILES***Convert CAS to SMILES*

---

**Description**

This is a wrapper for `webchem::cir_query`, using the CACTUS API at [https://cactus.nci.nih.gov/chemical/structure\\_documentation](https://cactus.nci.nih.gov/chemical/structure_documentation) for the conversion. Before converting the CAS number, the name is checked whether it contains the word 'derivative'. If so, the conversion is stopped and NA is returned. Also, a warning will be printed in this case.

**Usage**

```
CAS2SMILES(CAS_number, name)
```

**Arguments**

<code>CAS_number</code>	character The CAS registry number of a compound
<code>name</code>	character The compound's name

**Details**

The API allows only one query per second. This is a hard-coded feature

**Value**

The SMILES code of the compound as character-string

**Author(s)**

pstahlhofen

**Examples**

```
SMILES_ethanol <- CAS2SMILES("64-17-5", "Ethanol")
```

---

**checkIsotopes***Checks for isotopes in a msmsWorkspace*

---

**Description**

Checks for isotopes in a `msmsWorkspace`

## Usage

```
checkIsotopes(
  w,
  mode = "pH",
  intensity_cutoff = 0,
  intensity_precision = "none",
  conflict = "strict",
  isolationWindow = 2,
  evalMode = "complete",
  plotSpectrum = TRUE,
  settings = getOption("RMassBank")
)
```

## Arguments

w	A <code>msmsWorkspace</code> to work with.
mode	"pH", "pNa", "pM", "pNH4", " $mH$ ", " $mM$ ", " $mFA$ " for different ions ( $[M+H]^+$ , $[M+Na]^+$ , $[M]^+$ , $[M+NH4]^+$ , $[M-H]^-$ , $[M]^-$ , $[M+FA]^-$ ).
intensity_cutoff	The cutoff (as an absolute intensity value) under which isotopic peaks shouldn't be checked for or accepted as valid. Please note: The cutoff is not hard in the sense that it interacts with the <code>intensity_precision</code> parameter.
intensity_precision	The difference that is accepted between the calculated and observed intensity of a possible isotopic peak. Further details down below.
conflict	Either "isotopic"(Peak formulas are always chosen if they fit the requirements for an isotopic peak) or "strict"(Peaks are only marked as isotopic when there hasn't been a formula assigned before.)
isolationWindow	Half of the width of the isolation window in Da
evalMode	Currently no function yet, but planned. Currently must be "complete"
plotSpectrum	A boolean specifying whether the spectrum should be plotted
settings	Options to be used for processing. Defaults to the options loaded via <code>loadRmbSettings</code> et al. Refer to there for specific settings.

## Details

text describing parameter inputs in more detail.

- `intensity_precision` This parameter determines how strict the intensity values should adhere to the calculated intensity in relation to the parent peak. Options for this parameter are "none", where the intensity is irrelevant, "low", which has an error margin of 70% and "high", where the error margin is set to 35%. The recommended setting is "low", but can be changed to adjust to the intensity precision of the mass spectrometer.

## Value

The `msmsWorkspace` with annotated isolation peaks

**Author(s)**

Michael Stravs, Eawag <michael.stravs@eawag.ch>

Erik Mueller, UFZ

---

checkSpectra

*Check if a spectra set is found, complete, empty*

---

**Description**

Checks if a specific compound (RmbSpectraSet) was found with child spectra in the raw file (found), has a complete set of MS2 spectra with useful peaks (complete), or is empty (note: spectra are currently not ever marked empty - empty should mean found, but no useful peaks at all. This is not yet currently tested.)

**Usage**

```
checkSpectra(s, property)

## S4 method for signature 'RmbSpectraSet,character'
checkSpectra(s, property)
```

**Arguments**

s	The (RmbSpectraSet) to check
property	The property to check (found, complete or empty)

**Value**

TRUE or FALSE

**Methods (by class)**

- checkSpectra(s = RmbSpectraSet, property = character):

**Author(s)**

stravsmi

---

cleanElnoise	<i>Remove electronic noise</i>
--------------	--------------------------------

---

## Description

Removes known electronic noise peaks from a peak table

## Usage

```
cleanElnoise(peaks, noise=getOption("RMassBank")$electronicNoise,
            width = getOption("RMassBank")$electronicNoiseWidth)

## S4 method for signature 'data.frame,numeric,numeric'
cleanElnoise(
  peaks,
  noise = getOption("RMassBank")$electronicNoise,
  width = getOption("RMassBank")$electronicNoiseWidth
)

## S4 method for signature 'RmbSpectrum2,numeric,numeric'
cleanElnoise(
  peaks,
  noise = getOption("RMassBank")$electronicNoise,
  width = getOption("RMassBank")$electronicNoiseWidth
)

## S4 method for signature 'RmbSpectrum2List,numeric,numeric'
cleanElnoise(
  peaks,
  noise = getOption("RMassBank")$electronicNoise,
  width = getOption("RMassBank")$electronicNoiseWidth
)

## S4 method for signature 'RmbSpectraSet,numeric,numeric'
cleanElnoise(
  peaks,
  noise = getOption("RMassBank")$electronicNoise,
  width = getOption("RMassBank")$electronicNoiseWidth
)
```

## Arguments

peaks	An aggregated peak frame as described in <a href="#">aggregateSpectra</a> . Columns <code>mzFound</code> , <code>dppm</code> and <code>dppmBest</code> are needed.
noise	A numeric vector of known m/z of electronic noise peaks from the instrument Defaults to the entries in the RMassBank settings.

width	The window for the noise peak in m/z units. Defaults to the entries in the RMass-Bank settings.
-------	---

### Value

Extends the aggregate data frame by column noise (logical), which is TRUE if the peak is marked as noise.

### Methods (by class)

- `cleanElnoise(peaks = data.frame, noise = numeric, width = numeric)`: Remove known electronic noise peaks
- `cleanElnoise(peaks = RmbSpectrum2, noise = numeric, width = numeric)`: Remove known electronic noise peaks
- `cleanElnoise(peaks = RmbSpectrum2List, noise = numeric, width = numeric)`: Remove known electronic noise peaks
- `cleanElnoise(peaks = RmbSpectraSet, noise = numeric, width = numeric)`: Remove known electronic noise peaks

### Author(s)

Michael Stravs

### See Also

[msmsWorkflow](#)

### Examples

```
# As used in the workflow:  
## Not run:  
  w@aggregated <-  
  cleanElnoise(w@aggregated)  
  
## End(Not run)
```

---

combineMultiplicities *Combine workspaces for multiplicity filtering*

---

### Description

Combines multiple msmsWorkspace items to one workspace which is used for multiplicity filtering.

### Usage

```
combineMultiplicities(workspaces)
```

## Arguments

workspaces	A vector of <code>msmsWorkspace</code> items. The first item is taken as the "authoritative" workspace, i.e. the one which will be used for the record generation. The subsequent workspaces will only be used for multiplicity filtering.
------------	--

## Details

This feature is particularly meant to be used in conjunction with the `confirmMode` option of `msmsWorkflow`: a file can be analyzed with `confirmMode = 0` (default) and subsequently with `confirmMode = 1` (take second highest scan). The second analysis should contain "the same" spectra as the first one (but less intense) and can be used to confirm the peaks in the first spectra.

TO DO: Enable the combination of workspaces for combining e.g. multiple energy settings measured separately.

## Value

A `msmsWorkspace` object prepared for step 8 processing.

## Author(s)

Stravs MA, Eawag <[michael.stravs@eawag.ch](mailto:michael.stravs@eawag.ch)>

## See Also

[msmsWorkspace-class](#)

## Examples

```
## Not run:
w <- newMsmsWorkspace
w@files <- c("spec1", "spec2")
w1 <- msmsWorkflow(w, steps=c(1:7), mode="pH")
w2 <- msmsWorkflow(w, steps=c(1:7), mode="pH", confirmMode = 1)
wTotal <- combineMultiplicities(c(w1, w2))
wTotal <- msmsWorkflow(wTotal, steps=8, mode="pH", archivename = "output")
# continue here with mbWorkflow

## End(Not run)
```

## Description

The resulting SDF will be written to a file named 'Compoundlist.sdf'. The header for each block is the chemical name, tags for ID, SMILES and CAS are added in the description block

**Usage**

```
compoundlist2SDF(filename)
```

**Arguments**

filename	character The name of the csv-file to be read. Note that the compoundlist has to be filtered already.
----------	---

**Value**

This method has no return value.

**Author(s)**

pstahlhofen

**Examples**

```
## Not run:  
compoundlist2SDF("Compoundlist_filtered.csv")  
  
## End(Not run)
```

---

createCompoundlist     *Create a Compoundlist from JCAMP files*

---

**Description**

This method will automatically look for all single-block JCAMP files in the directory by picking all files ending in '.dx' (and not '.jdx'). A csv-file named 'Compoundlist.csv' will be created in the same directory. The Compoundlist contains columns 'ID', 'Name', 'SMILES' and 'CAS' where 'SMILES' might be empty if the compound is a derivative or if the CAS number could not be converted (see CAS2SMILES).

**Usage**

```
createCompoundlist()
```

**Value**

This method has no return value.

**Author(s)**

pstahlhofen

**See Also**

CAS2SMILES

## Examples

```
## Not run:  
# Prepare the compoundlist-creation  
splitMultiblockDX('my_multiblock_jcamp.jdx')  
createCompoundlist()  
  
## End(Not run)
```

---

createMolfile

*Create MOL file for a chemical structure*

---

## Description

Creates a MOL file (in memory or on disk) for a compound specified by the compound ID or by a SMILES code.

## Usage

```
createMolfile(id_or_smiles, fileName = FALSE)
```

## Arguments

id_or_smiles	The compound ID or a SMILES code.
fileName	If the filename is set, the file is written directly to disk using the specified filename. Otherwise, it is returned as a text array.

## Details

The function invokes OpenBabel (and therefore needs a correctly set OpenBabel path in the RMassBank settings), using the SMILES code retrieved with `findSmiles` or using the SMILES code directly. The current implementation of the workflow uses the latter version, reading the SMILES code directly from the MassBank record itself.

## Value

A character array containing the MOL/SDF format file, ready to be written to disk.

## Author(s)

Michael Stravs

## References

OpenBabel: <http://openbabel.org>

## See Also

[findSmiles](#)

## Examples

```
# Benzene:  
## Not run:  
createMolfile("C1=CC=CC=C1")  
  
## End(Not run)
```

---

CTS.externalIdSubset    *Select a subset of external IDs from a CTS record.*

---

## Description

Select a subset of external IDs from a CTS record.

## Usage

```
CTS.externalIdSubset(data, database)
```

## Arguments

data	The complete CTS record as retrieved by <a href="#">getCtsRecord</a> .
database	The database for which keys should be returned.

## Value

Returns an array of all external identifiers stored in the record for the given database.

## Author(s)

Michele Stravs, Eawag <stravsmi@eawag.ch>

## Examples

```
## Not run:  
# Return all CAS registry numbers stored for benzene.  
data <- getCTSRecord("UHOVQNZJYSORNB-UHFFFAOYSA-N")  
cas <- CTS.externalIdSubset(data, "CAS")  
  
## End(Not run)
```

---

CTS.externalIdTypes     *Find all available databases for a CTS record*

---

### Description

Find all available databases for a CTS record

### Usage

CTS.externalIdTypes(data)

### Arguments

data                    The complete CTS record as retrieved by [getCTSRecord](#).

### Value

Returns an array of all database names for which there are external identifiers stored in the record.

### Author(s)

Michele Stravs, Eawag <stravsmi@eawag.ch>

### Examples

```
## Not run:
# Return all databases for which the benzene entry has
# links in the CTS record.

data <- getCTS("UHOVQNZJYSORNB-UHFFFAOYSA-N")
databases <- CTS.externalIdTypes(data)

## End(Not run)
```

---

dbe

*Calculate Double Bond Equivalents*

---

### Description

Calculates the Ring and Double Bond Equivalents for a chemical formula. The highest valence state of each atom is used, such that the returned DBE should never be below 0.

### Usage

dbe(formula)

**Arguments**

**formula** A molecular formula in text or list representation (e.g. "C6H12O6" or list(C=6, H=12, O=6) ).

**Value**

Returns the DBE for the given formula.

**Author(s)**

Michael Stravs

**Examples**

```
#  
dbe("C6H12O6")
```

---

deprofile

*De-profile a high-resolution MS scan in profile mode.*

---

**Description**

The deprofile functions convert profile-mode high-resolution input data to "centroid"-mode data amenable to i.e. centWave. This is done using full-width, half-height algorithm, spline algorithm or local maximum method.

**Usage**

```
deprofile.scan(scan, noise = NA, method = "deprofile.fwhm",  
              colnames = TRUE, ...)  
  
deprofile(df, noise, method, ...)  
  
deprofile.fwhm(df, noise = NA, cut = 0.5)  
  
deprofile.localMax(df, noise = NA)  
  
deprofile.spline(df, noise=NA, minPts = 5, step = 0.00001)
```

**Arguments**

**df** A data frame with at least the columns `mz` and `int` to perform deprofiling on.

**noise** The noise cutoff. A peak is not included if the maximum stick intensity of the peak is below the noise cutoff.

**method** "deprofile.fwhm" for full-width half-maximum or "deprofile.localMax" for local maximum.

...	Arguments to the workhorse functions <code>deprofile.fwhm</code> etc.
scan	A matrix with 2 columns for m/z and intensity; from profile-mode high-resolution data. Corresponds to the spectra obtained with <code>xcms::getScan</code> or <code>mzR::peaks</code> .
colnames	For <code>deprofile.scan</code> : return matrix with column names (xcms-style, TRUE, default) or plain (mzR-style, FALSE).
cut	A parameter for <code>deprofile.fwhm</code> indicating where the peak flanks should be taken. Standard is 0.5 (as the algorithm name says, at half maximum.) Setting <code>cut = 0.75</code> would instead determine the peak width at 3/4 maximum, which might give a better accuracy for merged peaks, but could be less accurate if too few data points are present.
minPts	The minimal points count in a peak to build a spline; for peaks with less points the local maximum will be used instead. Note: The minimum value is 4!
step	The interpolation step for the calculated spline, which limits the maximum precision which can be achieved.

## Details

The `deprofile.fwhm` method is basically an R-semantic version of the "Exact Mass" m/z deprofiler from MZmine. It takes the center between the m/z values at half-maximum intensity for the exact peak mass. The logic is stolen verbatim from the Java MZmine algorithm, but it has been rewritten to use the fast R vector operations instead of loops wherever possible. It's slower than the Java implementation, but still decently fast IMO. Using matrices instead of the data frame would be more memory-efficient and also faster, probably.

The `deprofile.localMax` method uses local maxima and is probably the same used by e.g. Xcalibur. For well-formed peaks, "deprofile.fwhm" gives more accurate mass results; for some applications, `deprofile.localMax` might be better (e.g. for fine isotopic structure peaks which are not separated by a valley and also not at half maximum.) For MS2 peaks, which have no isotopes, `deprofile.fwhm` is probably the better choice generally.

`deprofile.spline` calculates the mass using a cubic spline, as the HiRes peak detection in OpenMS does.

The word "centroid" is used for convenience to denote not-profile-mode data. The data points are NOT mathematical centroids; we would like to have a better word for it.

The `noise` parameter was only included for completeness, I personally don't use it.

`deprofile.fwhm` and `deprofile.localMax` are the workhorses; `deprofile.scan` takes a 2-column scan as input. `deprofile` dispatches the call to the appropriate worker method.

## Value

`deprofile.scan`: a matrix with 2 columns for m/z and intensity

## Note

Known limitations: If the absolute leftmost stick or the absolute rightmost stick in a scan are maxima, they will be discarded! However, I don't think this will ever present a practical problem.

**Author(s)**

Michael Stravs, Eawag <michael.stravs@eawag.ch>

**References**

mzMine source code [http://sourceforge.net/svn/?group\\_id=139835](http://sourceforge.net/svn/?group_id=139835)

**Examples**

```
## Not run:  
mzrFile <- openMSfile("myfile.mzML")  
acqNo <- xraw@acquisitionNum[[50]]  
scan.mzML.profile <- mzR::peaks(mzrFile, acqNo)  
scan.mzML <- deprofile.scan(scan.mzML.profile)  
close(mzrFile)  
  
## End(Not run)
```

---

**exportMassbank**

*Export internally stored MassBank data to files*

---

**Description**

Exports MassBank recfile data arrays and corresponding molfiles to physical files on hard disk, for one compound.

**Usage**

```
exportMassbank(compiled, molfile = NULL)
```

**Arguments**

compiled	RmbSpectraSet the spectra of one compound for which files should be exported
molfile	A molfile from <a href="#">createMolfile</a> ; deprecated since molfiles are not used by MassBank anymore.

**Details**

The data from `compiled` is still used here, because it contains the "visible" accession number. In the plain-text format contained in `files`, the accession number is not "accessible" anymore since it's in the file.

**Value**

No return value.

**Note**

An improvement would be to write the accession numbers into `names(compiled)` and later into `names(files)` so `compiled` wouldn't be needed here anymore. (The compound ID would have to go into `names(molfile)`, since it is also retrieved from `compiled`.)

**Author(s)**

Michael Stravs

**References**

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

**See Also**

[createMolfile](#), [toMassbank](#), [mbWorkflow](#)

---

fillback

*Fill back reanalyzed / refiltered peak info into spectra*

---

**Description**

This method takes the info which is added to the aggregated table in the reanalysis and multiplicity filtering steps of the workflow, and adds it back into the spectra.

**Usage**

```
fillback(o, id, aggregated)

## S4 method for signature 'msmsWorkspace,missing,missing'
fillback(o)

## S4 method for signature 'RmbSpectraSet,missing,data.frame'
fillback(o, aggregated)

## S4 method for signature 'RmbSpectrum2,character,data.frame'
fillback(o, id, aggregated)
```

**Arguments**

- o msmsWorkspace, RmbSpectraSet or RmbSpectrum2 The object information is filled back into. If applied to an RmbSpectraSet, information is added to all its RmbSpectrum2 children. If applied to the whole msmsWorkspace, information is added to all SpectraSets.
- id character or missing The id of the parent RmbSpectraSet if applied to RmbSpectrum2
- aggregated data.frame or missing The aggregated table of the parent msmsWorkspace if applied to RmbSpectraSet or RmbSpectrum2

**Value**

o msmsWorkspace, RmbSpectraSet or Rmbspectrum2 The same object that was given as input with new information filled into it

---

`filterCompoundlist`

*Filter a Compoundlist for missing SMILES values*

---

**Description**

Read the Compoundlist given by the filename and write a 'Compoundlist\_filtered.csv', containing only the lines with a SMILES string

**Usage**

```
filterCompoundlist(filename)
```

**Arguments**

filename character The name of the csv-file to be read

**Value**

This method has no return value.

**Author(s)**

pstahlhofen

**Examples**

```
## Not run:  
filterCompoundlist('Compoundlist.csv')  
  
## End(Not run)
```

---

filterLowaccResults    *Filter peaks with low accuracy*

---

## Description

Filters a peak table (with annotated formulas) for accuracy. Low-accuracy peaks are removed.

## Usage

```
filterLowaccResults(peaks, mode="fine", filterSettings = getOption("RMassBank")$filterSettings)
```

## Arguments

peaks            A data frame with at least the columns `mzFound` and `dppm`.  
mode            coarse or fine, see below.  
filterSettings   Settings for filtering. For details, see documentation of [analyzeMsMs](#)

## Details

In the coarse mode, mass tolerance is set to 10 ppm (above m/z 120) and 15 ppm (below m/z 120). This is useful for formula assignment before recalibration, where a wide window is desirable to accomodate the high mass deviations at low m/z values, so we get a nice recalibration curve.

In the fine run, the mass tolerance is set to 5 ppm over the whole mass range. This should be applied after recalibration.

## Value

A `list`(`TRUE` = `goodPeakDataframe`, `FALSE` = `badPeakDataframe`) is returned: A data frame with all peaks which are "good" is in `return[["TRUE"]]`.

## Author(s)

Michael Stravs

## See Also

[analyzeMsMs](#), [filterPeakSatellites](#)

## Examples

```
# from analyzeMsMs:  
## Not run: childPeaksFilt <- filterLowaccResults(childPeaksInt, filterMode)
```

---

**filterMultiplicity**      *filterMultiplicity*

---

**Description**

Multiplicity filtering: Removes peaks which occur only once in a n-spectra set.

**Usage**

```
filterMultiplicity(w, archivename=NA, mode="pH", recalcbest = TRUE,  
multiplicityFilter =getOption("RMassBank")$multiplicityFilter)
```

**Arguments**

w	Workspace containing the data to be processed (aggregate table and RmbSpectraSet objects)
archivename	The archive name, used for generation of archivename_Failpeaks.csv
mode	Mode of ion analysis
recalcbest	Boolean, whether to recalculate the formula multiplicity after the first multiplicity filtering step. Sometimes, setting this to FALSE can be a solution if you have many compounds with e.g. fluorine atoms, which often have multiple assigned formulas per peak and might occasionally lose peaks because of that.
multiplicityFilter	Threshold for the multiplicity filter. If set to 1, no filtering will apply (minimum 1 occurrence of peak). 2 equals minimum 2 occurrences etc.

**Details**

This function executes multiplicity filtering for a set of spectra using the workhorse function [filterPeaksMultiplicity](#) (see details there) and retrieves problematic filtered peaks (peaks which are of high intensity but were discarded, because either no formula was assigned or it was not present at least 2x), using the workhorse function [problematicPeaks](#). The results are returned in a format ready for further processing with [mbWorkflow](#).

**Value**

A list object with values:

peaksOK	Peaks with >1-fold formula multiplicity from the "normal" peak analysis.
peaksReanOK	Peaks with >1-fold formula multiplicity from peak reanalysis.
peaksFiltered	All peaks with annotated formula multiplicity from first analysis.
peaksFilteredReanalysis	All peaks with annotated formula multiplicity from peak reanalysis.
peaksProblematic	Peaks with high intensity which do not match inclusion criteria -> possible false negatives. The list will be exported into archivename_failpeaks.csv.

**Author(s)**

Michael Stravs

**See Also**

[filterPeaksMultiplicity](#), [problematicPeaks](#)

**Examples**

```
## Not run:
refilteredRcSpecs <- filterMultiplicity(
  w, "myarchive", "pH")

## End(Not run)
```

**filterPeakSatellites** *Filter satellite peaks*

**Description**

Filters satellite peaks in FT spectra which arise from FT artifacts and from conversion to stick mode. A very simple rule is used which holds mostly true for MSMS spectra (and shouldn't be applied to MS1 spectra which contain isotope structures...)

**Usage**

```
filterPeakSatellites(peaks, filterSettings = getOption("RMassBank")$filterSettings)
```

**Arguments**

<code>peaks</code>	A peak data frame with at least the columns <code>mz</code> , <code>int</code> . Note that <code>mz</code> is used even for the recalibrated spectra, i.e. the desatellited spectrum is identical for both the unrecalibrated and the recalibrated spectra.
<code>filterSettings</code>	The settings used for filtering. Refer to <a href="#">analyzeMsMs</a> documentation for filter settings.

**Details**

The function cuts off all peaks within 0.5 m/z from every peak, in decreasing intensity order, which are below 5 intensity. E.g. for peaks m/z=100, int=100; m/z=100.2, int=2, m/z=100.3, int=6, m/z 150, int=10: The most intense peak (m/z=100) is selected, all neighborhood peaks below 5 peak) and the next less intense peak is selected. Here this is the m/z=150 peak. All low-intensity neighborhood peaks are removed (nothing). The next less intense peak is selected (m/z=100.3) and again neighborhood peaks are cut away (nothing to cut here). Note that the m/z = 100.2 peak was already removed.)

**Value**

Returns the peak table with satellite peaks removed.

**Note**

This is a very crude rule, but works remarkably well for our spectra.

**Author(s)**

Michael Stravs

**See Also**

[analyzeMsMs](#), [filterLowaccResults](#)

**Examples**

```
# From the workflow:  
## Not run:  
  # Filter out satellite peaks:  
  shot <- filterPeakSatellites(shot)  
  shot_satellite_n <- setdiff(row.names(shot_full), row.names(shot))  
  shot_satellite <- shot_full[shot_satellite_n,]  
  # shot_satellite contains the peaks which were eliminated as satellites.  
  
## End(Not run)
```

---

**filterPeaksMultiplicity**

*Multiplicity filtering: Removes peaks which occur only once in a n-spectra set.*

---

**Description**

For every compound, every peak (with annotated formula) is compared across all spectra. Peaks whose formula occurs only once for all collision energies / spectra types, are discarded. This eliminates "stochastic formula hits" of pure electronic noise peaks efficiently from the spectra. Note that in the author's experimental setup two spectra were recorded at every collision energy, and therefore every peak-formula should appear at least twice if it is real, even if it is by chance a fragment which appears on only one collision energy setting. The function was not tested in a different setup. Therefore, use with a bit of caution.

**Usage**

```
filterPeaksMultiplicity(w, recalcbest = TRUE)
```

## Arguments

w	a ‘msmsWorkspace’ object where formulas have been assigned to peaks
recalcBest	Whether the best formula for each peak should be re-determined. This is necessary for results from the ordinary <code>analyzeMsMs</code> analysis which allows multiple potential formulas per peak - the old best match could potentially have been dropped because of multiplicity filtering. For results from <code>reanalyzeFailpeak</code> this is not necessary, since only one potential formula is assigned in this case.

## Value

The peak table is returned, enriched with columns:

- `formulaMultiplicity` The # of occurrences of this formula in the spectra of its compounds.

## Author(s)

Michael Stravs, EAWAG <michael.stravs@eawag.ch>

## Examples

```
## Not run:
peaksFiltered <- filterPeaksMultiplicity(peaksMatched(w),
  "formula", TRUE)
peaksOK <- subset(peaksFiltered, formulaMultiplicity > 1)

## End(Not run)
```

---

findEIC

*Extract EICs*

---

## Description

Extract EICs from raw data for a determined mass window.

## Usage

```
findEIC(
  msRaw,
  mz,
  limit = NULL,
  rtLimit = NA,
  headerCache = NULL,
  floatingRecalibration = NULL,
  peaksCache = NULL,
  polarity = NA,
  msLevel = 1,
  precursor = NULL
)
```

**Arguments**

msRaw	The mzR file handle
mz	The mass or mass range to extract the EIC for: either a single mass (with the range specified by limit below) or a mass range in the form of c(min, max).
limit	If a single mass was given for mz: the mass window to extract. A limit of 0.001 means that the EIC will be returned for [mz - 0.001, mz + 0.001].
rtLimit	If given, the retention time limits in form c(rtmin, rtmax) in seconds.
headerCache	If present, the complete mzR::header(msRaw). Passing this value is useful if spectra for multiple compounds should be extracted from the same mzML file, since it avoids getting the data freshly from msRaw for every compound.
floatingRecalibration	A fitting function that predict()s a mass shift based on the retention time. Can be used if a lockmass calibration is known (however you have to build the calibration yourself.)
peaksCache	If present, the complete output of mzR::peaks(msRaw). This speeds up the lookup if multiple compounds should be searched in the same file.
polarity	If a value is given, scans are filtered to this polarity before EIC extraction. Valid values are 1 for positive, 0 for negative (according to mzR), or a RMassBank 'mode' (e.g. 'pH, pM, mH...') from which the polarity can be derived.
msLevel	Which MS level to target for EIC extraction. By default this is 1; level 2 can be used to extract the EIC of fragments for a specific precursor or to extract EICs from DIA data.
precursor	Which precursor to target for EIC extraction. If 'NULL', the scans are not filtered by precursor. Use this only for 'msLevel != 1'. If a 'precursor' filter is set for 'msLevel == 1', all scans will be filtered out.

**Value**

A [rt, intensity, scan] matrix (scan being the scan number.)

**Author(s)**

Michael A. Stravs, Eawag <michael.stravs@eawag.ch>

**See Also**

[findMsMsHR](#)

---

**findMass***Calculate exact mass*

---

**Description**

Retrieves the exact mass of the uncharged molecule. It works directly from the SMILES and therefore is used in the MassBank workflow ([mbWorkflow](#)) - there, all properties are calculated from the SMILES code retrieved from the database. (Alternatively, takes also the compound ID as parameter and looks it up.) Calculation relies on Rcdk.

**Usage**

```
findMass(cpdID_or_smiles, retrieval = "standard", mode = "pH")
```

**Arguments**

cpdID_or_smiles	SMILES code or compound ID of the molecule. (Numerics are treated as compound ID).
retrieval	A value that determines whether the files should be handled either as "standard", if the compoundlist is complete, "tentative", if at least a formula is present or "unknown" if the only known thing is the m/z
mode	"pH", "pNa", "pM", "pNH4", "mH", "mM", "mFA" for different ions ([M+H]+, [M+Na]+, [M]+, [M+NH4]+, [M-H]-, [M]-, [M+FA]-). Only needed for retrieval="unknown"

**Value**

Returns the exact mass of the uncharged molecule.

**Author(s)**

Michael Stravs

**See Also**

[findMz](#)

**Examples**

```
##  
findMass("OC[C@H]1OC(O)[C@H](O)[C@@H](O)[C@@H]1O")
```

---

`findMsMsHR`

*Extract MS/MS spectra for specified precursor*

---

## Description

Extracts MS/MS spectra from LC-MS raw data for a specified precursor, specified either via the RMassBank compound list (see [loadList](#)) or via a mass.

## Usage

```
findMsMsHR(  
  fileName = NULL,  
  msRaw = NULL,  
  cpdID,  
  mode = "pH",  
  confirmMode = 0,  
  useRtLimit = TRUE,  
  ppmFine = getOption("RMassBank")$findMsMsRawSettings$ppmFine,  
  mzCoarse = getOption("RMassBank")$findMsMsRawSettings$mzCoarse,  
  fillPrecursorScan = getOption("RMassBank")$findMsMsRawSettings$fillPrecursorScan,  
  rtMargin = getOption("RMassBank")$rtMargin,  
  deprofile = getOption("RMassBank")$deprofile,  
  headerCache = NULL,  
  peaksCache = NULL,  
  enforcePolarity = getOption("RMassBank")$enforcePolarity,  
  diaWindows = getOption("RMassBank")$findMsMsRawSettings$diaWindows  
)  
  
findMsMsHR.mass(  
  msRaw,  
  mz,  
  limit.coarse,  
  limit.fine,  
  rtLimits = NA,  
  maxCount = NA,  
  headerCache = NULL,  
  fillPrecursorScan = FALSE,  
  deprofile = getOption("RMassBank")$deprofile,  
  peaksCache = NULL,  
  cpdID = NA,  
  polarity = NA,  
  diaWindows = getOption("RMassBank")$findMsMsRawSettings$diaWindows  
)
```

## Arguments

`fileName` The file to open and search the MS2 spectrum in.

msRaw	The opened raw file (mzR file handle) to search the MS2 spectrum in. Specify either this or <code>fileName</code> .
cpdID	The compound ID in the compound list (see <a href="#">loadList</a> ) to use for formula lookup. Note: In <code>\code{findMsMsHR.mass}</code> , this is entirely optional and used only in case a warning must be displayed; compound lookup is done via mass only.
mode	The processing mode (determines which ion/adduct is searched): "pH", "pNa", "pM", "pNH4", "mH", "mM", "mFA" for different ions ([M+H]+, [M+Na]+, [M]+, [M+NH4]+, [M-H]-, [M]-, [M+FA]-).
confirmMode	Whether to use the highest-intensity precursor (=0), second- highest (=1), third-highest (=2)...
useRtLimit	Whether to respect retention time limits from the compound list.
ppmFine	The limit in ppm to use for fine limit (see below) calculation.
mzCoarse	The coarse limit to use for locating potential MS2 scans: this tolerance is used when finding scans with a suitable precursor ion value.
fillPrecursorScan	If TRUE, the precursor scan will be filled from MS1 data. To be used for data where the precursor scan is not stored in the raw data.
rtMargin	The retention time tolerance to use.
deprofile	Whether deprofiling should take place, and what method should be used (cf. <a href="#">deprofile</a> )
headerCache	If present, the complete <code>mzR::header(msRaw)</code> . Passing this value is useful if spectra for multiple compounds should be extracted from the same mzML file, since it avoids getting the data freshly from <code>msRaw</code> for every compound.
peaksCache	If present, the complete output of <code>mzR::peaks(msRaw)</code> . This speeds up the lookup if multiple compounds should be searched in the same file.
enforcePolarity	If TRUE, scans are filtered for the given ‘mode’’s polarity when finding the target spectrum.
diaWindows	A data frame with columns <code>precursorMz</code> , <code>mzMin</code> , <code>mzMax</code> which specifies the precursor and window size of each window for DIA acquisition.
mz	The mass to use for spectrum search.
limit.coarse	Parameter in <code>findMsMsHR.mass</code> corresponding to <code>mzCoarse</code> . (The parameters are distinct to clearly conceptually distinguish <code>findMsMsHR.mass</code> (a standalone useful function) from the <code>cpdID</code> based functions (workflow functions).)
limit.fine	The fine limit to use for locating MS2 scans: this tolerance is used when locating an appropriate analyte peak in the MS1 precursor spectrum.
rtLimits	<code>c(min, max)</code> : Minimum and maximum retention time to use when locating the MS2 scans.
maxCount	The maximal number of spectra groups to return. One spectra group consists of all data-dependent scans from the same precursor whose precursor mass matches the specified search mass.
polarity	If set (for <code>?findMsMsHR.mass</code> ), scans are filtered for the given ‘mode’’s polarity when finding the target spectrum.

## Details

Different versions of the function get the data from different sources. Note that `findMsMsHR` and `findMsMsHR.direct` differ mainly in that `findMsMsHR` opens a file whereas `findMsMs.direct` uses an open file handle - both are intended to be used in a full process which involves compound lists etc. In contrast, `findMsMsHR.mass` is a low-level function which uses the mass directly for lookup and is intended for use as a standalone function in unrelated applications.

## Value

An `RmbSpectraSet` (for `findMsMsHR`). Contains parent MS1 spectrum (`@parent`), a block of dependent MS2 spectra (`@children`) and some metadata (`id,mz,name,mode` in which the spectrum was acquired).

For `findMsMsHR.mass`: a list of `RmbSpectraSets` as defined above, sorted by decreasing precursor intensity.

## Functions

- `findMsMsHR.mass()`: A submethod of `findMsMsHR` that retrieves basic spectrum data

## Note

`findMsMs.direct` is deactivated

## Author(s)

Michael A. Stravs, Eawag <[michael.stravs@eawag.ch](mailto:michael.stravs@eawag.ch)>

## See Also

`findEIC`

## Examples

```
## Not run:
loadList("mycompoundlist.csv")
# if Atrazine has compound ID 1:
msms_atrazine <- findMsMsHR(fileName = "Atrazine_0001_pos.mzML", cpdID = 1, mode = "pH")
# Or alternatively:
msRaw <- openMSfile("Atrazine_0001_pos.mzML")
msms_atrazine <- findMsMsHR(msRaw=msRaw, cpdID = 1, mode = "pH")
# Or directly by mass (this will return a list of spectra sets):
mz <- findMz(1)$mzCenter
msms_atrazine_all <- findMsMsHR.mass(msRaw, mz, 1, ppm(msRaw, 10, p=TRUE))
msms_atrazine <- msms_atrazine_all[[1]]

## End(Not run)
```

---

findMsMsHR.direct      *Discontinued: find MS/MS spectrum from open raw file*

---

## Description

This interface has been discontinued. [findMsMsHR](#) now supports the same parameters (use named parameters).

## Usage

```
findMsMsHR.direct(  
  msRaw,  
  cpdID,  
  mode = "pH",  
  confirmMode = 0,  
  useRtLimit = TRUE,  
  ppmFine = getOption("RMassBank")$findMsMsRawSettings$ppmFine,  
  mzCoarse = getOption("RMassBank")$findMsMsRawSettings$mzCoarse,  
  fillPrecursorScan = getOption("RMassBank")$findMsMsRawSettings$fillPrecursorScan,  
  rtMargin = getOption("RMassBank")$rtMargin,  
  deprofile = getOption("RMassBank")$deprofile,  
  headerCache = NULL  
)
```

## Arguments

msRaw	x
cpdID	x
mode	x
confirmMode	x
useRtLimit	x
ppmFine	x
mzCoarse	x
fillPrecursorScan	x
rtMargin	x
deprofile	x
headerCache	x

## Value

an error

## Author(s)

stravsmi

---

findMsMsHR.ticms2	<i>Extract an MS/MS spectrum from MS2 TIC</i>
-------------------	---

---

## Description

Extract an MS/MS spectrum or multiple MS/MS spectra based on the TIC of the MS2 and precursor mass, picking the most intense MS2 scan. Can be used, for example, to get a suitable MS2 from direct infusion data which was collected with purely targeted MS2 without MS1.

## Usage

```
findMsMsHR.ticms2(
  msRaw,
  mz,
  limit.coarse,
  limit.fine,
  rtLimits = NA,
  maxCount = NA,
  headerCache = NULL,
  fillPrecursorScan = FALSE,
  deprofile = getOption("RMassBank")$deprofile,
  trace = "ms2tic"
)
```

## Arguments

msRaw	The mzR raw file
mz	Mass to find
limit.coarse	Allowed mass deviation for scan precursor (in m/z values)
limit.fine	Unused here, but present for interface compatibility with findMsMsHR
rtLimits	Unused here, but present for interface compatibility with findMsMsHR
maxCount	Maximal number of spectra to return
headerCache	Cached results of header(msRaw), either to speed up the operations or to operate with preselected header() data
fillPrecursorScan	Unused here, but present for interface compatibility with findMsMsHR
deprofile	Whether deprofiling should take place, and what method should be used (cf. <a href="#">deprofile</a> )
trace	Either "ms2tic" or "ms2basepeak": Which intensity trace to use - can be either the TIC of the MS2 or the basepeak intensity of the MS2.

## Details

Note that this is not a precise function and only really makes sense in direct infusion and if the precursor is really known, because MS2 precursor data is only "roughly" accurate (to 2 dp). The regular `findMsMsHR` functions confirm the exact mass of the precursor in the MS1 scan.

**Value**

a list of "spectrum sets" as defined in [findMsMsHR](#), sorted by decreasing precursor intensity.

**Author(s)**

stravsmi

---

<code>findMsMsHRperMsp</code>	<i>Retrieve spectra from msp files</i>
-------------------------------	--

---

**Description**

This function is currently used to read msp files containing data that were already processed in order to convert the results to MassBank records.

**Usage**

```
findMsMsHRperMsp(fileName, cpdIDs, mode = "pH")
findMsMsHRperMsp.direct(fileName, cpdIDs, mode = "pH")
```

**Arguments**

<code>fileName</code>	vector of character-strings The msp files to be searched for spectra
<code>cpdIDs</code>	vector of integers The IDs of compounds in the compoundlist for which spectra should be retrieved
<code>mode</code>	character, default: "pH" The processing mode that was used to produce the spectrum. Should be one of "pH": ([M+H]+) "pNa": ([M+Na]+) "pM": ([M]+) "mH": ([M-H]-) or "mFA": ([M+FA]-) (see the RMassBank vignette)

**Value**

An `RmbSpectraSet` with integrated information from the msp files

**Functions**

- `findMsMsHRperMsp.direct()`: A submethod of `findMsMsHRperxcms` that retrieves basic spectrum data

---

findMsMsHRperxcms *Read in mz-files using XCMS*

---

## Description

Picks peaks from mz-files and returns the pseudospectra that CAMERA creates with the help of XCMS

## Usage

```
findMsMsHRperxcms(  
  fileName,  
  cpdID,  
  mode = "pH",  
  findPeaksArgs = NULL,  
  plots = FALSE,  
  MSe = FALSE  
)  
  
findMsMsHRperxcms.direct(  
  fileName,  
  cpdID,  
  mode = "pH",  
  findPeaksArgs = NULL,  
  plots = FALSE,  
  MSe = FALSE  
)
```

## Arguments

fileName	The path to the mz-file that should be read
cpdID	The compoundID(s) of the compound that has been used for the file
mode	The ionization mode that has been used for the spectrum represented by the peaklist
findPeaksArgs	A list of arguments that will be handed to the xcms-method findPeaks via do.call
plots	A parameter that determines whether the spectra should be plotted or not
MSe	A boolean value that determines whether the spectra were recorded using MSe or not

## Value

The spectra generated from XCMS

## Functions

- `findMsMsHRperxcms.direct()`: A submethod of `findMsMsHRperxcms` that retrieves basic spectrum data

**Author(s)**

Erik Mueller

**See Also**

[msmsWorkflow toRMB](#)

**Examples**

```
## Not run:  
fileList <- list.files(system.file("XCMSinput", package = "RMassBank"), "Glucolesquerellin", full.names=TRUE)[3]  
loadList(system.file("XCMSinput/compoundList.csv", package="RMassBank"))  
psp <- findMsMsHrperxcms(fileList,2184)  
  
## End(Not run)
```

---

findMz

*Find compound information*

---

**Description**

Retrieves compound information from the loaded compound list or calculates it from the SMILES code in the list.

**Usage**

```
findMz(cpdID, mode = "pH", ppm = 10, deltaMz = 0, retrieval="standard",  
unknownMass =getOption("RMassBank")$unknownMass )  
  
findRt(cpdID)  
  
findSmiles(cpdID)  
  
findFormula(cpdID, retrieval="standard")  
  
findCAS(cpdID)  
  
findName(cpdID)  
  
findLevel(cpdID, compact=FALSE)
```

**Arguments**

cpdID                   The compound ID in the compound list.

mode	Specifies the species of the molecule: An empty string specifies uncharged monoisotopic mass, <i>pH</i> (positive H) specifies [M+H]+, <i>pNa</i> specifies [M+Na]+, <i>pM</i> specifies [M]+, <i>mH</i> and <i>mFA</i> specify [M-H]- and [M+FA]-, respectively. (I apologize for the naming of <i>pH</i> which has absolutely nothing to do with chemical <i>pH</i> values.)
ppm	Specifies ppm window (10 ppm will return the range of the molecular mass + and - 10 ppm).
deltaMz	Specifies additional m/z window to add to the range (deltaMz = 0.02 will return the range of the molecular mass +- 0.02 (and additionally +- the set ppm value).
retrieval	A value that determines whether the files should be handled either as "standard", if the compoundlist is complete, "tentative", if at least a formula is present or "unknown" if the only known thing is the m/z
unknownMass	'charged' or 'neutral' ('charged' assumed by default) specifies whether a mass of an unknown compound (level 5) refers to the charged or neutral mass (and correspondingly, whether it must be shifted or not to find the m/z value)
compact	Only for <i>findLevel</i> , returns the "retrieval" parameter used for many functions within RMassBank if TRUE

### Value

*findMz* will return a `list(mzCenter=, mzMin=, mzMax=)` with the molecular weight of the given ion, as calculated from the SMILES code and Rcdk.

*findRt*, *findSmiles*, *findCAS*, *findName* will return the corresponding entry from the compound list. *findFormula* returns the molecular formula as determined from the SMILES code.

### Author(s)

Michael Stravs

### See Also

[findMass](#), [loadList](#), [findMz.formula](#)

### Examples

```
## Not run: %
findMz(123, "pH", 5)
findFormula(123)

## End(Not run)
```

---

**findMz.formula** *Find the exact mass +/- a given margin for a given formula or its ions and adducts.*

---

## Description

Find the exact mass +/- a given margin for a given formula or its ions and adducts.

## Usage

```
findMz.formula(formula, mode = "pH", ppm = 10, deltaMz = 0)
```

## Arguments

formula	The molecular formula in text or list format (see <a href="#">formulastring.to.list</a> )
mode	"pH", "pNa", "pM", "pNH4", "mH", "mM", "mFA" for different ions ([M+H]+, [M+Na]+, [M]+, [M+NH4]+, [M-H]-, [M]-, [M+FA]-). "" for the uncharged molecule.
ppm	The ppm margin to add/subtract
deltaMz	The absolute mass to add/subtract. Cumulative with ppm

## Value

A list(mzMin=, mzCenter=, mzMax=) with the masses.

## Author(s)

Michael A. Stravs, Eawag <michael.stravs@eawag.ch>

## See Also

[findMz](#)

## Examples

```
findMz.formula("C6H6")
```

---

findProgress	<i>Determine processed steps</i>
--------------	----------------------------------

---

## Description

This function reads out the content of different slots of the workspace object and finds out which steps have already been processed on it.

## Usage

```
findProgress(workspace)
```

## Arguments

workspace      A msmsWorkspace object.

## Value

An array containing all msmsWorkflow steps which have likely been processed.

## Author(s)

Stravs MA, Eawag <michael.stravs@eawag.ch>

## Examples

```
## Not run:  
findProgress(w)  
  
## End(Not run)
```

---

flatten	<i>Flatten, or re-read, MassBank header blocks</i>
---------	--

---

## Description

flatten converts a list of MassBank compound information sets (as retrieved by [gatherData](#)) to a flat table, to be exported into an [infolist](#). readMbdata reads a single record from an infolist flat table back into a MassBank (half-)entry.

## Usage

```
flatten(mbdata)  
  
readMbdata(row)
```

## Arguments

mbdata	A list of MassBank compound information sets as returned from <a href="#">gatherData</a> .
row	One row of MassBank compound information retrieved from an infolist.

## Details

Neither the flattening system itself nor the implementation are particularly fantastic, but since hand-checking of records is a necessary evil, there is currently no alternative (short of coding a complete GUI for this and working directly on the records.)

## Value

`flatten` returns a tibble (not a data frame or matrix) to be written to CSV.

`readMbdata` returns a list of type `list(id= compoundID, ..., 'ACCESSION' = '', 'RECORD_TITLE' = '', )` etc.

## Author(s)

Michael Stravs

## References

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

## See Also

[gatherData](#), [loadInfolist](#)

## Examples

```
## Not run:
# Collect some data to flatten
ids <- c(40,50,60,70)
data <- lapply(ids, gatherData)
# Flatten the data trees to a table
flat.table <- flatten(data)
# reimport the table into a tree
data.reimported <- apply(flat.table, 1, readMbdata)

## End(Not run)
```

---

**formulastring.to.list** *Interconvert molecular formula representations*

---

**Description**

Converts molecular formulas from string to list representation or vice versa.

**Usage**

```
list.to.formula(flist)  
  
formulastring.to.list(formula)
```

**Arguments**

formula	A molecular formula in string format, e.g. "C6H12O6".
flist	A molecular formula in list format, e.g. list( "C" = 6, "H" = 12, "O" = 6 ).

**Details**

The function doesn't care about whether your formula makes sense. However, "C3.504" will give list("C" = 3, "O" = 4) because regular expressions are used for matching (however, list("C" = 3.5, "O" = 4) gives "C3.504".) Duplicate elements cause problems; only "strict" molecular formulas ("CH4O", but not "CH3OH") work correctly.

**Value**

`list.to.formula` returns a string representation of the formula; `formulastring.to.list` returns the list representation.

**Author(s)**

Michael Stravs

**See Also**

[add.formula](#), [order.formula](#), [is.valid.formula](#)

**Examples**

```
#  
list.to.formula(list("C" = 4, "H" = 12))  
# This is also OK and useful to calculate e.g. adducts or losses.  
list.to.formula(list("C" = 4, "H" = -1))  
formulastring.to.list(list.to.formula(formulastring.to.list("CHIBr"))))
```

---

**gatherCCTE***Retrieve supplemental annotation data from US EPA*

---

**Description**

Retrieves annotation data for a compound from the internet service US EPA CCTE based on the inchikey generated by babel or Cactus

**Usage**

```
gatherCCTE(key, api_key)
```

**Arguments**

key	An Inchi-Key or other chemical identifier (e.g. Chemical name, DTXSID, CASRN, InChIKey, DTXCID)
api_key	An US EPA CCTE API key (personal or application)

**Details**

The data retrieved is the US EPA DTXSID, the US EPA chemical dashboard substance ID, the CAS-RN, the DTX preferred name, and the DTXCID (chemical ID).

**Value**

Returns a list with 5 slots: dtxsid The US EPA chemical dashboard substance id dtxcid The US EPA chemical dashboard chemical id preferredName The US EPA chemical dashboard preferred name casrn The latest CAS registration number smiles The SMILES annotation of the structure

**Author(s)**

Tobias Schulze

**References**

CCTE REST: <https://api-ccte.epa.gov/docs/>

**See Also**

[mbWorkflow](#)

**Examples**

```
# Gather data for compound ID 131
## Not run: gatherCCTE("QEIXBXXKTUNWDK-UHFFFAOYSA-N", api_key = NA)
```

---

gatherData	<i>Retrieve annotation data</i>
------------	---------------------------------

---

## Description

Retrieves annotation data for a compound from the internet services CTS, Pubchem, Chemspider and Cactvs, based on the SMILES code and name of the compounds stored in the compound list.

## Usage

```
gatherData(id)
```

## Arguments

id	The compound ID.
----	------------------

## Details

Composes the "upper part" of a MassBank record filled with chemical data about the compound: name, exact mass, structure, CAS no., links to PubChem, KEGG, ChemSpider. The instrument type is also written into this block (even if not strictly part of the chemical information). Additionally, index fields are added at the start of the record, which will be removed later: id, dbcas, dbname from the compound list, dataused to indicate the used identifier for CTS search (smiles or dbname).

Additionally, the fields ACCESSION and RECORD\_TITLE are inserted empty and will be filled later on.

## Value

Returns a list of type `list(id=compoundID, ..., 'ACCESSION' = '', 'RECORD_TITLE' = '', )` etc.

## Author(s)

Michael Stravs

## References

Chemical Translation Service: <http://uranus.fiehnlab.ucdavis.edu:8080/cts/homePage> cactus Chemical Identifier Resolver: <http://cactus.nci.nih.gov/chemical/structure> MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf) Pubchem REST: [https://pubchem.ncbi.nlm.nih.gov/pug\\_rest/PUG\\_REST.html](https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST.html) Chemspider InChI conversion: <https://www.chemspider.com/InChI.asmx>

## See Also

[mbWorkflow](#)

## Examples

```
# Gather data for compound ID 131
## Not run: gatherData(131)
```

---

gatherDataBabel	<i>Retrieve annotation data</i>
-----------------	---------------------------------

---

## Description

Retrieves annotation data for a compound by using babel, based on the SMILES code and name of the compounds stored in the compound list.

## Usage

```
gatherDataBabel(id)
```

## Arguments

id	The compound ID.
----	------------------

## Details

Composes the "upper part" of a MassBank record filled with chemical data about the compound: name, exact mass, structure, CAS no.. The instrument type is also written into this block (even if not strictly part of the chemical information). Additionally, index fields are added at the start of the record, which will be removed later: id, dbcas, dbname from the compound list.

Additionally, the fields ACCESSION and RECORD\_TITLE are inserted empty and will be filled later on.

This function is an alternative to gatherData, in case CTS is down or if information on one or more of the compounds in the compound list are sparse

## Value

Returns a list of type `list(id=compoundID, ..., 'ACCESSION' = '', 'RECORD_TITLE' = '', )` etc.

## Author(s)

Michael Stravs, Erik Mueller

## References

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

## See Also

[mbWorkflow](#)

## Examples

```
# Gather data for compound ID 131
## Not run: gatherDataBabel(131)
```

---

gatherDataUnknown	<i>Retrieve annotation data</i>
-------------------	---------------------------------

---

## Description

Retrieves annotation data for an unknown compound by using basic information present

## Usage

```
gatherDataUnknown(id, mode, retrieval)
```

## Arguments

id	The compound ID.
mode	"pH", "pNa", "pM", "pNH4", "mH", "mM", "mFA" for different ions ([M+H]+, [M+Na]+, [M]+, [M+NH4]+, [M-H]-, [M]-, [M+FA]-).
retrieval	A value that determines whether the files should be handled either as "standard", if the compoundlist is complete, "tentative", if at least a formula is present or "unknown" if the only known thing is the m/z

## Details

Composes the "upper part" of a MassBank record filled with chemical data about the compound: name, exact mass, structure, CAS no.. The instrument type is also written into this block (even if not strictly part of the chemical information). Additionally, index fields are added at the start of the record, which will be removed later: id, dbcas, dbname from the compound list.

Additionally, the fields ACCESSION and RECORD\_TITLE are inserted empty and will be filled later on.

This function is used to generate the data in case a substance is unknown, i.e. not enough information is present to derive anything about formulas or links

## Value

Returns a list of type `list(id= compoundID, ..., 'ACCESSION' = ' ', 'RECORD_TITLE' = ' ', )` etc.

## Author(s)

Michael Stravs, Erik Mueller

## References

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

## See Also

[mbWorkflow](#)

## Examples

```
# Gather data for compound ID 131
## Not run: gatherDataUnknown(131, "pH")
```

---

gatherPubChem

*Retrieve supplemental annotation data from Pubchem*

---

## Description

Retrieves annotation data for a compound from the internet service Pubchem based on the inchikey generated by babel or Cactus

## Usage

```
gatherPubChem(key)
```

## Arguments

key	An Inchi-Key
-----	--------------

## Details

The data retrieved is the Pubchem CID, a synonym from the Pubchem database, the IUPAC name (using the preferred if available) and a Chebi link

## Value

Returns a list with 4 slots: **PcID** The Pubchem CID **Synonym** An arbitrary synonym for the compound name **IUPAC** A IUPAC-name (preferred if available) **Chebi** The identification number of the chebi database

## Author(s)

Erik Mueller

## References

Pubchem REST: [https://pubchem.ncbi.nlm.nih.gov/pug\\_rest/PUG\\_REST.html](https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST.html) Chebi: <http://www.ebi.ac.uk/chebi>

**See Also**[mbWorkflow](#)**Examples**

```
# Gather data for compound ID 131
## Not run: gatherPubChem("QEIXBXXKTUNWDK-UHFFFAOYSA-N")
```

---

getAnalyticalInfo      *Get analytical info for MassBank record*

---

**Description**

Collects the info for ‘ai, ac\_lc, ac\_ms‘ for general, LC and MS info respectively. The info comes from the settings except for the compound-specific part, which is omitted if there is no ‘cpd‘ specified.

**Usage**

```
getAnalyticalInfo(cpd = NULL)
```

**Arguments**

cpd      A ‘RmbSpectraSet‘ object

---

getCactus      *Retrieve information from Cactus*

---

**Description**

Retrieves information from the Cactus Chemical Identifier Resolver (PubChem).

**Usage**

```
getCactus(identifier, representation)
```

**Arguments**

identifier      Any identifier interpreted by the resolver, e.g. an InChI key or a SMILES code.  
representation      The desired representation, as required from the resolver. e.g. stdinchikey, chemspider\_id, formula... Refer to the webpage for details.

**Details**

It is not necessary to specify in which format the `identifier` is. Somehow, cactus does this automatically.

### Value

The result of the query, in plain text. Can be NA, or one or multiple lines (character array) of results.

### Note

Note that the InChI key is retrieved with a prefix (InChIkey=), which must be removed for most database searches in other databases (e.g. CTS).

### Author(s)

Michael Stravs

### References

cactus Chemical Identifier Resolver: <http://cactus.nci.nih.gov/chemical/structure>

### See Also

[getCtsRecord](#), [getPcId](#)

### Examples

```
# Benzene:  
getCactus("C1=CC=CC=C1", "cas")  
getCactus("C1=CC=CC=C1", "stdinchikey")  
getCactus("C1=CC=CC=C1", "chemspider_id")
```

---

## getCASRN

*Search CCTE CAS registration number*

---

### Description

Retrieves CCTE CASRN from US EPA for a search term.

### Usage

```
getCASRN(key, api_key)
```

### Arguments

key	ID to be converted
api_key	API key for CCTE

### Details

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

**Value**

The CCTE CAS RN (in string type)

**Author(s)**

Tobias Schulze

**References**

CCTE search: <https://api-ccte.epa.gov/docs>

CCTE REST: <https://api-ccte.epa.gov/docs>

**Examples**

```
## Not run:  
getCASRN("MKXZASYAUGDDCJ-NJAFHUGGSA-N")  
  
## End(Not run)
```

---

getCSID

*Retrieve the Chemspider ID for a given compound*

---

**Description**

Given an InChIKey, this function queries the chemspider web API to retrieve the Chemspider ID of the compound with that InChIkey.

**Usage**

```
getCSID(query)
```

**Arguments**

query	The InChIKey of the compound
-------	------------------------------

**Value**

Returns the chemspide

**Author(s)**

Michele Stravs, Eawag <stravsmi@eawag.ch>

Erik Mueller, UFZ <erik.mueller@ufz.de>

## Examples

```
## Not run:
# Return all CAS registry numbers stored for benzene.
data <- getCTSRecord("UHOVQNZJYSORNB-UHFFFAOYSA-N")
cas <- CTS.externalIdSubset(data, "CAS")

## End(Not run)
```

### getCTSKey

*Convert a single ID to another using CTS.*

## Description

Convert a single ID to another using CTS.

## Usage

```
getCTSKey(query, from = "Chemical Name", to = "InChIKey")
```

## Arguments

query	ID to be converted
from	Type of input ID
to	Desired output ID

## Value

An unordered array with the resulting converted key(s).

## Author(s)

Michele Stravs, Eawag <stravsmi@eawag.ch>

## Examples

```
k <- getCTSKey("benzene", "Chemical Name", "InChIKey")
```

---

getCTSRecord	<i>Retrieve information from CTS</i>
--------------	--------------------------------------

---

## Description

Retrieves a complete CTS record from the InChI key.

## Usage

```
getCTSRecord(key)
```

## Arguments

key	The InChI key.
-----	----------------

## Value

Returns a list with all information from CTS: `inchicode`, `formula`, `exactmass` contain single values. `synonyms` contains an unordered list of scored synonyms (`type`, `name`, `score`, where `type` indicates either a normal name or a specific IUPAC name, see below). `externalIDs` contains an unordered list of identifiers of the compound in various databases (`name`, `value`, where `name` is the database name and `value` the identifier in that database.)

## Note

Currently, the CTS results are still incomplete; the name scores are all 0, formula and exact mass return zero.

## Author(s)

Michele Stravs, Eawag <stravsmi@eawag.ch>

## References

Chemical Translation Service: <https://cts.fiehnlab.ucdavis.edu>

## Examples

```
data <- getCTSRecord("UHOVQNZJYSORNB-UHFFFAOYSA-N")
# show all synonym "types"
types <- unique(unlist(lapply(data$synonyms, function(i) i$type)))
## Not run: print(types)
```

---

getData	<i>Get data frame with all present peak data</i>
---------	--

---

### Description

Returns a data frame with columns for all non-empty slots in a `RmbSpectrum2` object. Note that `MSnbase::Spectrum` has a method `as.data.frame`, however that one will return only `mz`, `intensity`. This function is kept separate to ensure downwards compatibility since it returns more columns than `MSnbase as.data.frame`.

### Usage

```
## S4 method for signature 'RmbSpectrum2'
getData(s)
```

### Arguments

`s` The `RmbSpectrum2` object to extract data from.

### Value

A data frame with columns for every set slot.

### Author(s)

stravsmi

---

getDTXCID	<i>Search CCTE DTXCID</i>
-----------	---------------------------

---

### Description

Retrieves CCTE DTXCID from US EPA for a search term.

### Usage

```
getDTXCID(key, api_key)
```

### Arguments

<code>key</code>	ID to be converted
<code>api_key</code>	API key for CCTE

### Details

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

**Value**

The DTXCID (in string type)

**Author(s)**

Tobias Schulze

**References**

CCTE search: <https://api-ccte.epa.gov/docs>

CCTE REST: <https://api-ccte.epa.gov/docs>

**Examples**

```
## Not run:  
getDTXID("MKXZASYAUGDDCJ-NJAFHUGGSA-N")  
  
## End(Not run)
```

---

getDTXSID

*Search CCTE DTXSID*

---

**Description**

Retrieves CCTE DTXSID from US EPA for a search term.

**Usage**

```
getDTXSID(key, api_key)
```

**Arguments**

key	ID to be converted
api_key	API key for CCTE

**Details**

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

**Value**

The DTXSID (in string type)

**Author(s)**

Tobias Schulze

## References

CCTE search: <https://api-ccte.epa.gov/docs>  
CCTE REST: <https://api-ccte.epa.gov/docs>

## Examples

```
## Not run:  
getDTXSID("MKXZASYAUGDDCJ-NJAFHUGGSA-N")  
  
## End(Not run)
```

---

getDTXSMILES                    *Search CCTE SMILES*

---

## Description

Retrieves CCTE SMILES from US EPA for a search term.

## Usage

```
getDTXSMILES(key, api_key)
```

## Arguments

key	ID to be converted
api_key	API key for CCTE

## Details

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

## Value

The SMILES (in string type)

## Author(s)

Tobias Schulze

## References

CCTE search: <https://api-ccte.epa.gov/docs>  
CCTE REST: <https://api-ccte.epa.gov/docs>

## Examples

```
## Not run:  
getDTXSMILES("MKXZASYAUGDDCJ-NJAFHUGGSA-N")  
  
## End(Not run)
```

---

getField

*Get the content of a field in a JCAMP file*

---

## Description

The content will always be returned as character-string

## Usage

```
getField(parsedJDX, field_name)
```

## Arguments

parsedJDX	list as created by readJDX A parsed, single-block JCAMP file
field_name	character The name of the field (e.g. 'CAS REGISTRY NO')

## Value

The field's content

## Author(s)

pstahlhofen

## See Also

readJDX

## Examples

```
## Not run:  
parsedJDX <- readJDX('my_singleblock_jcamp.dx')  
title <- getField(parsedJDX, "TITLE")  
  
## End(Not run)
```

---

**getMolecule***Create Rcdk molecule from SMILES*

---

**Description**

Generates a Rcdk molecule object from SMILES code, which is fully typed and usable (in contrast to the built-in `parse.smiles`).

**Usage**

```
getMolecule(smiles)
```

**Arguments**

smiles	The SMILES code of the compound.
--------	----------------------------------

**Details**

**NOTE: As of today (2012-03-16), Rcdk discards stereochemistry when loading the SMILES code!** Therefore, do not trust this function blindly, e.g. don't generate InChI keys from the result. It is, however, useful if you want to compute the mass (or something else) with Rcdk.

**Value**

A Rcdk IAtomContainer reference.

**Author(s)**

Michael Stravs

**See Also**

[parse.smiles](#)

**Examples**

```
# Lindane:  
getMolecule("C1(C(C(C(C1C1)C1)C1)C1)C1")  
# Benzene:  
getMolecule("C1=CC=CC=C1")
```

---

getPcId	<i>Search Pubchem CID</i>
---------	---------------------------

---

**Description**

Retrieves PubChem CIDs for a search term.

**Usage**

```
getPcId(query, from = "inchickey")
```

**Arguments**

query	ID to be converted
from	Type of input ID

**Details**

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

**Value**

The PubChem CID (in string type).

**Author(s)**

Michael Stravs, Erik Mueller

**References**

PubChem search: <http://pubchem.ncbi.nlm.nih.gov/>

Pubchem REST: [https://pubchem.ncbi.nlm.nih.gov/pug\\_rest/PUG\\_REST.html](https://pubchem.ncbi.nlm.nih.gov/pug_rest/PUG_REST.html)

**See Also**

[getCtsRecord](#), [getCactus](#)

**Examples**

```
getPcId("MKXZASYAUGDDCJ-NJAFHUGGSA-N")
```

---

getPrefName	<i>Search CCTE Preferred Name</i>
-------------	-----------------------------------

---

## Description

Retrieves CCTE Preferred Name from US EPA for a search term.

## Usage

```
getPrefName(key, api_key)
```

## Arguments

key	ID to be converted
api_key	API key for CCTE

## Details

Only the first result is returned currently. **The function should be regarded as experimental and has not thoroughly been tested.**

## Value

The CCTE Preferred Name (in string type)

## Author(s)

Tobias Schulze

## References

CCTE search: <https://api-ccte.epa.gov/docs>

CCTE REST: <https://api-ccte.epa.gov/docs>

## Examples

```
## Not run:  
getPrefName("MKXZASYAUGDDCJ-NJAFHUGGSA-N")  
  
## End(Not run)
```

---

is.valid.formula	<i>Check validity of formula</i>
------------------	----------------------------------

---

## Description

Checks whether the formula is chemically valid, i.e. has no zero-count or negative-count elements.

## Usage

```
is.valid.formula(formula)
```

## Arguments

formula	A molecular formula in string or list representation ("C6H6" or list(C=6,H=6)).
---------	---

## Details

The check is only meant to identify formulas which have negative elements, which can arise from the subtraction of adducts. It is **not** a high-level formula "validity" check like e.g. the Rcdk function `isinvalid.formula` which uses the nitrogen rule or a DBE rule.

## Author(s)

Michael Stravs

## See Also

[list.to.formula](#), [add.formula](#), [order.formula](#)

## Examples

```
#  
is.valid.formula(list(C=0,H=1,Br=2))  
is.valid.formula("CH2Cl")  
is.valid.formula("C0H2")
```

---

loadInfolists	<i>Load MassBank compound information lists</i>
---------------	---

---

## Description

Loads MassBank compound information lists (i.e. the lists which were created in the first two steps of the MassBank [mbWorkflow](#) and subsequently edited by hand.).

## Usage

```
loadInfolists(mb, path)

loadInfolist(mb, fileName)

resetInfolists(mb)
```

## Arguments

mb	The mbWorkspace to load/reset the lists in.
path	Directory in which the namelists reside. All CSV files in this directory will be loaded.
fileName	A single namelist to be loaded.

## Details

`resetInfolists` clears the information lists, i.e. it creates a new empty list in `mbdata_archive`. `loadInfolist` loads a single CSV file, whereas `loadInfolists` loads a whole directory.

## Value

The new workspace with loaded/reset lists.

## Author(s)

Michael Stravs, Tobias Schulze

## Examples

```
#
## Not run: mb <- resetInfolists(mb)
mb <- loadInfolist(mb, "my_csv_infolist.csv")
## End(Not run)
```

---

loadList	<i>Load compound list for RMassBank</i>
----------	---

---

## Description

Loads a CSV compound list with compound IDs

## Usage

```
loadList(path, listEnv=NULL, check=TRUE)  
  
resetList()
```

## Arguments

path	Path to the CSV list.
listEnv	The environment to load the list into. By default, the namelist is loaded into an environment internally in RMassBank.
check	A parameter that specifies whether the SMILES-Codes in the list should be checked for readability by rcdk.

## Details

The list is loaded into the variable *compoundList* in the environment *listEnv* (which defaults to the global environment) and used by the *findMz*, *findCAS*, ... functions. The CSV file is required to have at least the following columns, which are used for further processing and must be named correctly (but present in any order): *ID*, *Name*, *SMILES*, *RT*, *CAS*

*resetList()* clears a currently loaded list.

## Value

No return value.

## Author(s)

Michael Stravs

## See Also

[findMz](#)

## Examples

```
##  
## Not run: loadList("mylist.csv")
```

---

`makeMollist`*Write list.tsv file*

---

**Description**

Makes a list.tsv file in the "moldata" folder.

**Usage**

```
makeMollist(compiled)
```

**Arguments**

compiled	list of RmbSpectraSet compiled spectra for multiple compounds (one RmbSpectraSet each).
----------	---

**Details**

Generates the list.tsv file which is needed by MassBank to connect records with their respective molfiles. The first compound name is linked to a mol-file with the compound ID (e.g. 2334.mol for ID 2334).

**Value**

No return value.

**Author(s)**

Michael A. Stravs, Eawag <michael.stravs@eawag.ch>

---

`makePeaksCache`*Generate peaks cache*

---

**Description**

Generates a peak cache table for use with [findMsMsHR](#) functions.

**Usage**

```
makePeaksCache(msRaw, headerCache)
```

**Arguments**

msRaw	the input raw datafile (opened)
headerCache	the cached header, or subset thereof for which peaks should be extracted. Peak extraction goes by seqNum.

**Value**

A list of dataframes as from `mzR::peaks`.

**Author(s)**

stravsmi

---

makeRecalibration	<i>Recalibrate MS/MS spectra</i>
-------------------	----------------------------------

---

**Description**

Recalibrates MS/MS spectra by building a recalibration curve of the assigned putative fragments of all spectra in `aggregatedSpecs` (measured mass vs. mass of putative associated fragment) and additionally the parent ion peaks.

**Usage**

```
makeRecalibration(w,
  recalibrateBy =getOption("RMassBank")$recalibrateBy,
  recalibrateMS1 =getOption("RMassBank")$recalibrateMS1,
  recalibrator =getOption("RMassBank")$recalibrator,
  recalibrateMS1Window =getOption("RMassBank")$recalibrateMS1Window
  )

recalibrateSpectra(rawspec = NULL, rc = NULL, rc.ms1=NULL, w = NULL,
  recalibrateBy =getOption("RMassBank")$recalibrateBy,
  recalibrateMS1 =getOption("RMassBank")$recalibrateMS1)

recalibrateSingleSpec(spectrum, rc,
  recalibrateBy =getOption("RMassBank")$recalibrateBy)
```

**Arguments**

<code>w</code>	For <code>makeRecalibration</code> : to perform the recalibration with. For <code>recalibrateSpectra</code> : the <code>msmsWorkspace</code> which contains the recalibration curves (alternatively to specifying <code>rc</code> , <code>rc.ms1</code> ).
<code>recalibrateBy</code>	Whether recalibration should be done by ppm ("ppm") or by m/z ("mz").
<code>recalibrateMS1</code>	Whether MS1 spectra should be recalibrated separately ("separate"), together with MS2 ("common") or not at all ("none"). Usually taken from settings.
<code>recalibrator</code>	The recalibrator functions to be used. Refer to <code>recalibrate</code> for details. Usually taken from settings.

recalibrateMS1Window	Window width to look for MS1 peaks to recalibrate (in ppm).
spectrum	For recalibrateSingleSpec: a MSnbase Spectrum-derived object, commonly a RmbSpectrum2 for MS2 or Spectrum1 for MS1.
rawspec	For recalibrateSpectra: an RmbSpectraSetList of RmbSpectraSet objects, as the w@spectra slot from msmsWorkspace or any object returned by <a href="#">findMsMsHR</a> . If empty, no spectra are recalibrated, but the recalibration curve is returned.
rc, rc.ms1	The recalibration curves to be used in the recalibration.

## Details

Note that the actually used recalibration functions are governed by the general MassBank settings (see [recalibrate](#)).

If a set of acquired LC-MS runs contains spectra for two different ion types (e.g. [M+H]<sup>+</sup> and [M+Na]<sup>+</sup>) which should both be processed by RMassBank, it is necessary to do this in two separate runs. Since it is likely that one ion type will be the vast majority of spectra (e.g. most in [M+H]<sup>+</sup> mode), and only few spectra will be present for other specific adducts (e.g. only few [M+Na]<sup>+</sup> spectra), it is possible that too few spectra are present to build a good recalibration curve using only e.g. the [M+Na]<sup>+</sup> ions. Therefore we recommend, for one set of LC/MS runs, to build the recalibration curve for one ion type (msmsWorkflow(mode="pH", steps=c(1:8), newRecalibration=TRUE)) and reuse the same curve for processing different ion types (msmsWorkflow(mode="pNa", steps=c(1:8), newRecalibration=FALSE)). This also ensures a consistent recalibration across all spectra of the same batch.

## Value

makeRecalibration: a list(rc, rc.ms1) with recalibration curves for the MS2 and MS1 spectra.  
 recalibrateSpectra: if rawspec is not NULL, returns the recalibrated spectra as RmbSpectraSetList. All spectra have their mass recalibrated and evaluation data deleted.  
 recalibrateSingleSpec: the recalibrated Spectrum (same object, recalibrated masses, evaluation data like assigned formulae etc. deleted).

## Author(s)

Michael Stravs, Eawag <michael.stravs@eawag.ch>

## Examples

```
## Not run:
rcCurve <- recalibrateSpectra(w, "pH")
w@spectra <- recalibrateSpectra(mode="pH", rawspec=w@spectra, w=myWorkspace)
w@spectra <- recalibrateSpectra(mode="pH", rawspec=w@spectra, rcCurve$rc, rcCurve$rc.ms1)

## End(Not run)
```

---

**mbWorkflow***MassBank record creation workflow*

---

**Description**

Uses data generated by [msmsWorkflow](#) to create MassBank records.

**Usage**

```
mbWorkflow(  
  mb,  
  steps = c(1, 2, 3, 4, 5, 6, 7, 8),  
  infolist_path = "./infolist.csv",  
  gatherData = "online",  
  filter = TRUE  
)
```

**Arguments**

<code>mb</code>	The <code>mbWorkspace</code> to work in.
<code>steps</code>	Which steps in the workflow to perform.
<code>infolist_path</code>	A path where to store newly downloaded compound informations, which should then be manually inspected.
<code>gatherData</code>	A variable denoting whether to retrieve information using several online databases <code>gatherData= "online"</code> or to use the local babel installation <code>gatherData= "babel"</code> . Note that babel is used either way, if a directory is given in the settings. This setting will be ignored if retrieval is set to "standard"
<code>filter</code>	If TRUE, the peaks will be filtered according to the standard processing workflow in RMassBank - only the best formula for a peak is retained, and only peaks passing multiplicity filtering are retained. If FALSE, it is assumed that the user has already done filtering, and all peaks in the spectrum should be printed in the record (with or without formula.)

**Details**

See the vignette `vignette("RMassBank")` for detailed informations about the usage.

Steps:

Step 1: Find which compounds don't have annotation information yet. For these compounds, pull information from several databases (using `gatherData`).

Step 2: If new compounds were found, then export the `infolist.csv` and stop the workflow. Otherwise, continue.

Step 3: Take the archive data (in table format) and reformat it to MassBank tree format.

Step 4: Compile the spectra. Using the skeletons from the archive data, create MassBank records per compound and fill them with peak data for each spectrum. Also, assign accession numbers based on scan mode and relative scan no.

Step 5: Convert the internal tree-like representation of the MassBank data into flat-text string arrays (basically, into text-file style, but still in memory)

Step 6: For all OK records, generate a corresponding molfile with the structure of the compound, based on the SMILES entry from the MassBank record. (This molfile is still in memory only, not yet a physical file)

Step 7: If necessary, generate the appropriate subdirectories, and actually write the files to disk.

Step 8: Create the list.tsv in the molfiles folder, which is required by MassBank to attribute substances to their corresponding structure molfiles.

### Value

The processed mbWorkspace.

### Author(s)

Michael A. Stravs, Eawag <michael.stravs@eawag.ch>

### See Also

[mbWorkspace-class](#)

### Examples

```
## Not run:
mb <- newMbWorkspace(w) # w being a msmsWorkspace
mb <- loadInfolists(mb, "D:/myInfolistPath")
mb <- mbWorkflow(mb, steps=c(1:3), "newinfos.csv")

## End(Not run)
```

---

mbWorkspace-class      *Workspace for mbWorkflow data*

---

### Description

A workspace which stores input and output data for use with mbWorkflow.

### Usage

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

### Arguments

object      The mbWorkspace to display.

## Details

Slots:

**spectra, aggregated** The corresponding input data from [msmsWorkspace-class](#)

**additionalPeaks** A list of additional peaks which can be loaded using [addPeaks](#).

**mbdata, mbdata\_archive, mbdata\_relisted** Infolist data: Data for annotation of MassBank records, which can be loaded using [loadInfolists](#).

**compiled, compiled\_ok** Compiled tree-structured MassBank records. `compiled_ok` contains only the compounds with at least one valid spectrum.

**mbfiles** Compiled MassBank records in text representation.

**molfile** MOL files with the compound structures.

**ok,problems** Index lists for internal use which denote which compounds have valid spectra.

Methods:

**show** Shows a brief summary of the object. Currently only a stub.

## Author(s)

Michael Stravs, Eawag <michael.stravs@eawag.ch>

## See Also

[mbWorkflow](#)

---

mergePeaks

*Merge peaks for spectra merging, FT shoulder elimination etc.*

---

## Description

This procedure first sorts peaks by intensity (descending sort) and then starts iterating over the peaks, removing all entries that deviate "sufficiently far" from the currently selected peak. See the Details section for a full explanation and information on how to fine-tune peak removal.

## Usage

```
mergePeaks(peaks, ...)

## S4 method for signature 'data.frame'
mergePeaks(peaks, ...)

## S4 method for signature 'matrix'
mergePeaks(peaks, ...)

## S4 method for signature 'RmbSpectrum2'
mergePeaks(peaks, ...)

## S4 method for signature 'Spectrum'
mergePeaks(peaks, ...)
```

## Arguments

peaks	data.frame, matrix or RmbSpectrum2 The peak-table to be merged. In case of an RmbSpectrum2-object, peaks are retrieved and updated via <a href="#">getData</a> and <a href="#">setData</a> , respectively
...	3 numeric values These define cutoff limits (see details)

## Details

Three parameters must be passed to `mergePeaks` for peak-removal control in this order: - `cutoff_dppm_limit` - `cutoff_absolute_limit` - `cutoff_intensity_limit` The method iterates through the peaks, beginning with the highest-intensity peak and in each step removes all other peaks that fulfill conditions 1 AND 2 relative to the selected peak 1. Their  $m/z$  value does not deviate too far from the one of the selected peak. i.e. if the selected peak is  $p$  and the checked peak is  $c$ , it holds that EITHER  $|p.mz - c.mz| \leq \text{cutoff\_absolute\_limit}$  OR  $|p.mz - c.mz| \leq \text{ppm}(p.mz, \text{cutoff\_dppm\_limit}, p=\text{TRUE})$  (see [ppm](#)) 2. Their intensity is much smaller than the one of the selected peak, i.e.  $c.mz < \text{cutoff\_intensity\_limit} * p.mz$  for a suitable `cutoff_intensity_limit` between 0 and 1.

## Value

object of the same class as `peaks` The result contains a reduced peak-table ordered by  $m/z$

## See Also

[getData](#), [setData](#), [ppm](#)

## Examples

```
## Not run: mergePeaks(spectrum, 10, 0.5, 0.05)
```

---

mergeSpectra

*Merge multiple spectra into one*

---

## Description

This method takes a collection of RmbSpectrum2 objects and merges them into a single RmbSpectrum2 object

## Usage

```
mergeSpectra(spectra, ...)

## S4 method for signature 'RmbSpectrum2List'
mergeSpectra(spectra, ...)
```

**Arguments**

spectra	RmbSpectrum2List A list of RmbSpectrum2 objects to be merged
...	NOTHING (This parameter is reserved for future implementations of the generic)

**Details**

Information from all spectra is retrieved via [getData](#) combined with rbind and placed into the new spectrum with [setData](#)

**Value**

A single RmbSpectrum2 object containing the merged information

**See Also**

[getData](#), [setData](#)

---

msmsRead

*Extracts and processes spectra from a specified file list, according to loaded options and given parameters.*

---

**Description**

The filenames of the raw LC-MS runs are read from the array `files` in the global environment. See the vignette `vignette("RMassBank")` for further details about the workflow.

**Usage**

```
msmsRead(  
  w,  
  filetable = NULL,  
  files = NULL,  
  cpdids = NULL,  
  readMethod,  
  mode = NULL,  
  confirmMode = FALSE,  
  useRtLimit = TRUE,  
  Args = NULL,  
  settings = getOption("RMassBank"),  
  progressbar = "progressBarHook",  
  MSe = FALSE,  
  plots = FALSE  
)
```

**Arguments**

w	A <code>msmsWorkspace</code> to work with.
filetable	The path to a .csv-file that contains the columns "Files" and "ID" supplying the relationships between files and compound IDs. Either this or the parameter "files" need to be specified.
files	A vector or list containing the filenames of the files that are to be read as spectra. For the IDs to be inferred from the filenames alone, there need to be exactly 2 underscores.
cpdids	A vector or list containing the compound IDs of the files that are to be read as spectra. The ordering of this and <code>files</code> implicitly assigns each ID to the corresponding file. If this is supplied, then the IDs implicitly named in the filenames are ignored.
readMethod	Several methods are available to get peak lists from the files. Currently supported are "mzR", "xcms", "MassBank" and "peaklist". The first two read MS/MS raw data, and differ in the strategy used to extract peaks. MassBank will read existing records, so that e.g. a recalibration can be performed, and "peaklist" just requires a CSV with two columns and the column header "mz", "int".
mode	"pH", "pNa", "pM", "pNH4", "mH", "mM", "mFA" for different ions ([M+H]+, [M+Na]+, [M]+, [M+NH4]+, [M-H]-, [M]-, [M+FA]-). For 'readMethod == "mzR"', a vector of 'mode' entries is supported. The user should check that they are either all positive or negative. If this isn't the case, the recalibration will be incorrect.
confirmMode	Defaults to false (use most intense precursor). Value 1 uses the 2nd-most intense precursor for a chosen ion (and its data-dependent scans), etc.
useRtLimit	Whether to enforce the given retention time window.
Args	A list of arguments that will be handed to the xcms-method <code>findPeaks</code> via <code>do.call</code>
settings	Options to be used for processing. Defaults to the options loaded via <code>loadRmbSettings</code> et al. Refer to there for specific settings.
progressbar	The progress bar callback to use. Only needed for specialized applications. Cf. the documentation of <code>progressBarHook</code> for usage.
MSe	A boolean value that determines whether the spectra were recorded using MSe or not
plots	A boolean value that determines whether the pseudospectra in XCMS should be plotted

**Value**

The `msmsWorkspace` with msms-spectra read.

**Author(s)**

Michael Stravs, Eawag <michael.stravs@eawag.ch>

Erik Mueller, UFZ

**See Also**

[msmsWorkspace-class](#), [msmsWorkflow](#)

---

msmsRead.RAW

*Extracts and processes spectra from a list of xcms-Objects*

---

**Description**

The filenames of the raw LC-MS runs are read from the array files in the global environment. See the vignette `vignette("RMassBank")` for further details about the workflow.

**Usage**

```
msmsRead.RAW(
  w,
  xRAW = NULL,
  cpdids = NULL,
  mode,
  findPeaksArgs = NULL,
  settings = getOption("RMassBank"),
  progressbar = "progressBarHook",
  plots = FALSE
)
```

**Arguments**

w	A <code>msmsWorkspace</code> to work with.
xRAW	A list of <code>xcmsRaw</code> objects whose peaks should be detected and added to the workspace. The relevant data must be in the <code>MS1</code> data of the <code>xcmsRaw</code> object. You can coerce the <code>msn</code> -data in a usable object with the <code>msn2xcmsRaw</code> function of <code>xcms</code> .
cpdids	A vector or list containing the compound IDs of the files that are to be read as spectra. The ordering of this and <code>files</code> implicitly assigns each ID to the corresponding file. If this is supplied, then the IDs implicitly named in the filenames are ignored.
mode	" <code>pH</code> ", " <code>pNa</code> ", " <code>pM</code> ", " <code>pNH4</code> ", " <code>mH</code> ", " <code>mM</code> ", " <code>mFA</code> " for different ions ( $[M+H]^+$ , $[M+Na]^+$ , $[M]^+$ , $[M+NH4]^+$ , $[M-H]^-$ , $[M]^-$ , $[M+FA]^-$ ).
findPeaksArgs	A list of arguments that will be handed to the <code>xcms</code> -method <code>findPeaks</code> via <code>do.call</code>
settings	Options to be used for processing. Defaults to the options loaded via <code>loadRmbSettings</code> et al. Refer to there for specific settings.
progressbar	The progress bar callback to use. Only needed for specialized applications. Cf. the documentation of <code>progressBarHook</code> for usage.
plots	A boolean value that determines whether the pseudospectra in XCMS should be plotted

**Value**

The msmsWorkspace with msms-spectra read.

**Author(s)**

Michael Stravs, Eawag <michael.stravs@eawag.ch>

Erik Mueller, UFZ

**See Also**

[msmsWorkspace-class](#), [msmsWorkflow](#)

---

msmsWorkflow

*RMassBank mass spectrometry pipeline*

---

**Description**

Extracts and processes spectra from a specified file list, according to loaded options and given parameters.

**Usage**

```
msmsWorkflow(
  w,
  mode = "pH",
  steps = c(1:8),
  confirmMode = FALSE,
  newRecalibration = TRUE,
  useRtLimit = TRUE,
  archivename = NA,
  readMethod = "mzR",
  filetable = NULL,
  findPeaksArgs = NULL,
  plots = FALSE,
  precursorscan.cf = FALSE,
  settings = getOption("RMassBank"),
  analyzeMethod = "formula",
  progressbar = "progressBarHook",
  MSe = FALSE
)
```

**Arguments**

w	A msmsWorkspace to work with.
mode	"pH", "pNa", "pM", "mH", "mM", "mFA", "pNH4" for different ions ([M+H]+, [M+Na]+, [M]+, [M-H]-, [M]-, [M+FA]-, [M+NH4]+).

steps	Which steps of the workflow to process. See the vignette <code>vignette("RMassBank")</code> for details.
confirmMode	Defaults to false (use most intense precursor). Value 1 uses the 2nd-most intense precursor for a chosen ion (and its data-dependent scans), etc.
newRecalibration	Whether to generate a new recalibration curve (TRUE, default) or to reuse the currently stored curve (FALSE, useful e.g. for adduct-processing runs.)
useRtLimit	Whether to enforce the given retention time window.
archivename	The prefix under which to store the analyzed result files.
readMethod	Several methods are available to get peak lists from the files. Currently supported are "mzR", "xcms", "MassBank" and "peaklist". The first two read MS/MS raw data, and differ in the strategy used to extract peaks. MassBank will read existing records, so that e.g. a recalibration can be performed, and "peaklist" just requires a CSV with two columns and the column header "mz", "int".
filetable	If including step 1 (data extraction), a 'filetable' argument to be passed to <code>msmsRead</code> .
findPeaksArgs	A list of arguments that will be handed to the xcms-method <code>findPeaks</code> via <code>do.call</code>
plots	A parameter that determines whether the spectra should be plotted or not (This parameter is only used for the xcms-method)
precursorscan.cf	Whether to fill precursor scans. To be used with files which for some reasons do not contain precursor scan IDs in the mzML, e.g. AB Sciex converted files.
settings	Options to be used for processing. Defaults to the options loaded via <code>loadRmbSettings</code> et al. Refer to there for specific settings.
analyzeMethod	The "method" parameter to pass to <code>analyzeMsMs</code> .
progressbar	The progress bar callback to use. Only needed for specialized applications. Cf. the documentation of <code>progressBarHook</code> for usage.
MSe	A boolean value that determines whether the spectra were recorded using MSe or not

## Details

The filenames of the raw LC-MS runs are read from the array `files` in the global environment. See the vignette `vignette("RMassBank")` for further details about the workflow.

## Value

The processed `msmsWorkspace`.

## Author(s)

Michael Stravs, Eawag <[michael.stravs@eawag.ch](mailto:michael.stravs@eawag.ch)>

## See Also

[msmsWorkspace-class](#)

---

msmsWorkspace-class     *Workspace for msmsWorkflow data*

---

## Description

A workspace which stores input and output data for [msmsWorkflow](#).

## Usage

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

## Arguments

**object**     The msmsWorkspace to display.

## Details

Slots:

**files** The input file names

**spectra** The spectra per compound (RmbSpectraSet) extracted from the raw files

**aggregated** A data.frame with an aggregated peak table from all spectra. Further columns are added during processing.

**rc, rc.ms1** The recalibration curves generated in workflow step 4.

**parent** For the workflow steps after 4: the parent workspace containing the state (spectra, aggregate) before recalibration, such that the workflow can be reprocessed from start.

**archivename** The base name of the files the archive is stored to during the workflow.

**settings** The RMassBank settings used during the workflow, if stored with the workspace.#'

Methods:

**show** Shows a brief summary of the object and processing progress.

## Author(s)

Michael Stravs, Eawag <michael.stravs@eawag.ch>

## See Also

[msmsWorkflow](#)

---

newMbWorkspace	<i>Create new workspace for mbWorkflow</i>
----------------	--

---

### Description

Creates a new workspace for use with [mbWorkflow](#).

### Usage

```
newMbWorkspace(w)
```

### Arguments

w	The input <a href="#">msmsWorkspace</a> to load input data from.
---	--

### Details

The workspace input data will be loaded from the [msmsWorkspace-class](#) object provided by the parameter w.

### Value

A new [mbWorkflow](#) object with the loaded input data.

### Author(s)

Michael Stravs, Eawag <michael.stravs@eawag.ch>

### See Also

[mbWorkflow](#), [msmsWorkspace-class](#)

---

newMsmsWorkspace	<i>Create new empty workspace or load saved data for msmsWorkflow</i>
------------------	---

---

### Description

Creates an empty workspace or loads an existing workspace from disk.

### Usage

```
newMsmsWorkspace(files = character(0))
```

### Arguments

files	If given, the files list to initialize the workspace with.
-------	--

## Details

`newMsmsWorkspace` creates a new empty workspace for use with `msmsWorkflow`.

`loadMsmsWorkspace` loads a workspace saved using `archiveResults`. Note that it also successfully loads data saved with the old RMassBank data format into the new `msmsWorkspace` object.

## Value

A new `msmsWorkspace` object

## Author(s)

Michael Stravs, Eawag <michael.stravs@eawag.ch>

## See Also

`msmsWorkflow`, `msmsWorkspace-class`

---

### normalize, RmbSpectrum2-method

*Scale spectrum to specified intensity range*

---

## Description

Scale spectrum to specified intensity range

## Usage

```
## S4 method for signature 'RmbSpectrum2'
normalize(object, ..., scale = 999, precision = 0, slot = "intensity")
```

## Arguments

<code>object</code>	the 'RmbSpectrum2' object to scale
<code>...</code>	arguments passed to 'selectPeaks' to choose peaks for normalization
<code>scale</code>	Maximum intensity in normalized spectrum
<code>precision</code>	Digits after comma for normalized intensity, typically 0
<code>slot</code>	Which property of the spectrum should be scaled

---

normalize, RmbSpectrum2List-method  
*Normalize spectra*

---

**Description**

Scale all spectra in a ‘RmbSpectrum2List‘ to a specified intensity.

**Usage**

```
## S4 method for signature 'RmbSpectrum2List'  
normalize(object, ...)
```

**Arguments**

object	the ‘RmbSpectrum2List‘ with spectra to scale
...	Arguments passed to ‘normalize,RmbSpectrum2‘

---

order.formula      *Order a chemical formula correctly*

---

**Description**

Orders a chemical formula in the commonly accepted order (CH followed by alphabetic ordering).

**Usage**

```
order.formula(formula, as.formula = TRUE, as.list = FALSE)
```

**Arguments**

formula	A molecular formula in string or list representation ("C6H6" or list(C=6,H=6)).
as.formula	If TRUE, the return value is returned as a string. This is the default.
as.list	If TRUE, the return value is returned in list representation.

**Author(s)**

Michele Stravs

**See Also**

[list.to.formula](#), [add.formula](#), [is.valid.formula](#)

## Examples

```
#  
order.formula("H4C9")  
order.formula("C2N5HClBr")
```

---

parseMassBank

*MassBank-record Parser*

---

## Description

Can parse MassBank-records(only V2)

## Usage

```
parseMassBank(Files)
```

## Arguments

Files	array of character-strings Paths to the plaintext-records that should be read
-------	---

## Value

The mbWorkspace that the plaintext-record creates. All parsed information will be stored in the 'compiled\_ok' slot.

## Author(s)

Erik Mueller

## See Also

[validate](#)

## Examples

```
## Not run:  
paths <- c("filepath_to_records/RC000001.txt",  
        "filepath_to_records/RC000002.txt")  
mb <- parseMassBank(paths)  
  
## End(Not run)
```

---

parseMbRecord *MassBank-record Parser*

---

## Description

Can parse MassBank-records(only V2)

## Usage

```
parseMbRecord(filename, readAnnotation=TRUE)
```

## Arguments

filename	character A path to the plaintext-record that should be read
readAnnotation	logical, Default: TRUE If TRUE, parse annotations from the record file and add columns for 'formula', 'formulaCount', 'mzCalc' and 'dppm' to the peak table

## Value

An RmbSpectrum2 object created from the plaintext-record

## Author(s)

Erik Mueller

## See Also

[validate](#)

## Examples

```
## Not run:  
parseMassBank("filepath_to_records/RC00001.txt")  
## End(Not run)
```

peaksMatched

*Select matching/unmatching peaks from aggregate table***Description**

Select matching/unmatching peaks from aggregate table

**Usage**

```
peaksMatched(o)

## S4 method for signature 'data.frame'
peaksMatched(o)

## S4 method for signature 'msmsWorkspace'
peaksMatched(o)
```

**Arguments**

- o                   Workspace or aggregate table from a workspace

**Value**

Selects the peaks from the aggregate table which matched within filter criteria (peaksMatched) or didn't match (peaksUnmatched).

**Functions**

- peaksMatched(data.frame): A method to retrieve the matched peaks from the "aggregated" slot (a data.frame object) in an msmsWorkSpace
- peaksMatched(msmsWorkspace): A method to retrieve the matched peaks from an msmsWorkSpace

**Author(s)**

stravsmi

peaksUnmatched

*Select matching/unmatching peaks from aggregate table***Description**

Select matching/unmatching peaks from aggregate table

**Usage**

```
peaksUnmatched(o, cleaned = FALSE)

## S4 method for signature 'data.frame'
peaksUnmatched(o, cleaned = FALSE)

## S4 method for signature 'msmsWorkspace'
peaksUnmatched(o, cleaned = FALSE)
```

**Arguments**

- o                    Workspace or aggregate table from a workspace
- cleaned            Return only peaks which pass electronic noise filtering if TRUE.

**Value**

Selects the peaks from the aggregate table which matched within filter criteria (peaksMatched) or didn't match (peaksUnmatched).

**Methods (by class)**

- peaksUnmatched(data.frame): A method to retrieve the unmatched peaks from the "aggregated" slot (a data.frame object) in an msmsWorkSpace
- peaksUnmatched(msmsWorkspace): A method to retrieve the unmatched peaks from an msmsWorkSpace

**Author(s)**

stravsmi

---

plotMbWorkspaces	<i>Plots mbWorkspaces</i>
------------------	---------------------------

---

**Description**

Plots the peaks of one or two mbWorkspace to compare them.

**Usage**

```
plotMbWorkspaces(w1, w2 = NULL)
```

**Arguments**

- w1                    The mbWorkspace to be plotted
- w2                    Another optional mbWorkspace be plotted as a reference.

**Details**

This function plots one or two mbWorkspaces in case the user has used different methods to acquire similar spectra. w1 must always be supplied, while w2 is optional. The workspaces need to be fully processed for this function to work.

**Value**

A logical indicating whether the information was plotted or not

**Author(s)**

Erik Mueller

**Examples**

```
#  
## Not run: plotMbWorkspaces(w1,w2)
```

**plotRecalibration** *Plot the recalibration graph.*

**Description**

Plot the recalibration graph.

**Usage**

```
plotRecalibration(w, recalibrateBy =getOption("RMassBank")$recalibrateBy)  
  
plotRecalibration.direct(rcdata, rc, rc.ms1, title, mzrange,  
recalibrateBy =getOption("RMassBank")$recalibrateBy)
```

**Arguments**

w	The workspace to plot the calibration graph from
recalibrateBy	Whether recalibration was done by ppm ("ppm") or by m/z ("mz"). Important only for graph labeling here.
rcdata	A data frame with columns <code>recalfield</code> and <code>mzFound</code> .
rc	Predictor for MS2 data
rc.ms1	Predictor for MS1 data
title	Prefix for the graph titles
mzrange	m/z value range for the graph

**Author(s)**

Michele Stravs, Eawag <michael.stravs@eawag.ch>

---

ppm	<i>Calculate ppm values</i>
-----	-----------------------------

---

## Description

Calculates ppm values for a given mass.

## Usage

```
ppm(mass, dppm, l = FALSE, p = FALSE)
```

## Arguments

mass	The "real" mass
dppm	The mass deviation to calculate
l	Boolean: return limits? Defaults to FALSE.
p	Boolean: return ppm error itself? Defaults to FALSE.

## Details

This is a helper function used in RMassBank code.

## Value

By default (l=FALSE, p=FALSE) the function returns the mass plus the ppm error (for 123.00000 and 10 ppm: 123.00123, or for 123 and -10 ppm: 122.99877).

For l=TRUE, the function returns the upper and lower limit (sic!) For p=TRUE, just the difference itself is returned (0.00123 for 123/10ppm).

## Author(s)

Michael A. Stravs, Eawag <michael.stravs@eawag.ch>

## Examples

```
ppm(100, 10)
```

---

problematicPeaks	<i>Identify intense peaks (in a list of unmatched peaks)</i>
------------------	--

---

## Description

Finds a list of peaks in spectra with a high relative intensity ( $>10 \text{ 1e4}$ , or  $>1$  checked). Peaks orbiting around the parent peak mass (calculated from the compound ID), which are very likely co-isolated substances, are ignored.

## Usage

```
problematicPeaks(sp)
```

## Arguments

sp                    a RmbSpectrum2 object to be checked for problematic peaks.

## Value

The modified RmbSpectrum2 object with additional columns/properties ‘problematicPeaks‘ (logical ‘TRUE‘ if the peak is intense and unannotated), ‘aMax‘ (base peak intensity), ‘mzCenter‘ (the precursor m/z).

## Note

TODO: there is hardcoded logic in this function that needs to be resolved eventually!

## Author(s)

Michael Stravs

## See Also

[msmsWorkflow](#)

## Examples

# As used in the workflow:

```
sp <- new("RmbSpectrum2", mz = c(100,200,300,400,500), intensity = c(999999,888888,777777,666666,555555))
sp@ok <- TRUE
property(sp, "mzFound", addNew=TRUE) <- sp@mz
sp@good <- c(TRUE, TRUE, TRUE, FALSE, FALSE)
sp@precursorMz <- 600
sp_checked <- problematicPeaks(sp)
# stopifnot(sum(getData(sp_checked)$problematicPeak) == 2)
```

---

**processProblematicPeaks**

*Generate list of problematic peaks*

---

**Description**

Generates a list of intense unmatched peaks for further review (the "failpeak list") and exports it if the archive name is given.

**Usage**

```
processProblematicPeaks(w, archivename = NA)
```

**Arguments**

w	msmsWorkspace to analyze.
archivename	Base name of the archive to write to (for "abc" the exported failpeaks list will be "abc_Failpeaks.csv"). if the compoundlist is complete, "tentative", if at least a formula is present or "unknown" if the only known thing is the m/z

**Value**

Returns the aggregate data.frame with added column "problematic" (logical) which marks peaks which match the problematic criteria

**Author(s)**

stravsmi

---

**progressBarHook**

*Standard progress bar hook.*

---

**Description**

This function provides a standard implementation for the progress bar in RMassBank.

**Usage**

```
progressBarHook(object = NULL, value = 0, min = 0, max = 100, close = FALSE)
```

**Arguments**

object	An identifier representing an instance of a progress bar.
value	The new value to assign to the progress indicator
min	The minimal value of the progress indicator
max	The maximal value of the progress indicator
close	If TRUE, the progress bar is closed.

**Details**

RMassBank calls the progress bar function in the following three ways: `pb <- progressBarHook(object=NULL, value=0, min=0, max=LEN)` to create a new progress bar. `pb <- progressBarHook(object=pb, value=VAL)` to set the progress bar to a new value (between the set `min` and `max`) `progressBarHook(object=pb, close=TRUE)` to close the progress bar. (The actual calls are performed with `do.call`, e.g. `progressbar <- "progressBarHook"` `pb <- do.call(progressbar, list(object=pb, value= nProg))` . See the source code for details.)

To substitute the standard progress bar for an alternative implementation (e.g. for use in a GUI), the developer can write his own function which behaves in the same way as `progressBarHook`, i.e. takes the same parameters and can be called in the same way.

**Value**

Returns a progress bar instance identifier (i.e. an identifier which can be used as `object` in subsequent calls.)

**Author(s)**

Michele Stravs, Eawag <stravsmi@eawag.ch>

*property*

*Get a property of an RmbSpectrum2 object*

**Description**

This searches the 'properties' slot of the object and returns a column with matching name (if found) or `NULL` otherwise.

**Usage**

```
property(o, property)

## S4 method for signature 'RmbSpectrum2,character'
property(o, property)
```

**Arguments**

<code>o</code>	<code>RmbSpectrum2</code>
<code>property</code>	character The name of a property

**Value**

The corresponding column of `o@properties`

---

property<-	<i>Replacement function to set properties of an RmbSpectrum2 object</i>
------------	---

---

## Description

Update the 'properties' slot of the given object. If the column you want to update does not exist yet and addNew = FALSE (default), this will cause a warning and the object will not be changed

## Usage

```
property(o, property, addNew=FALSE, class="") <- value

## S4 replacement method for signature 'RmbSpectrum2,character,logical,character'
property(o, property, addNew = FALSE, class = "") <- value

## S4 replacement method for signature 'RmbSpectrum2,character,missing,character'
property(o, property, addNew = FALSE, class = "") <- value

## S4 replacement method for signature 'RmbSpectrum2,character,logical,missing'
property(o, property, addNew = FALSE, class = "") <- value

## S4 replacement method for signature 'RmbSpectrum2,character,missing,missing'
property(o, property, addNew = FALSE, class = "") <- value
```

## Arguments

o	RmbSpectrum2 The object whos 'properties' slot should be updated
property	character The name of the column in the 'properties' data frame to be updated
addNew	logical, Default: FALSE Whether or not a new column should be added in case a column of the given name does not exist yet.
class	character or missing The class of the entries for the column to be added/updated
value	ANY The value(s) to be written into the column

## Details

Please note that this is a replacement method, meaning that `property(o, property) <- value` can be used as a short-hand for the equivalent `o <- 'property<-'(o, property, value)`

## Value

The RmbSpectrum2 object with an updated 'properties' slot

---

reanalyzeFailpeaks	<i>Reanalyze unmatched peaks</i>
--------------------	----------------------------------

---

## Description

Reanalysis of peaks with no matching molecular formula by allowing additional elements (e.g. "N2O").

## Usage

```
reanalyzeFailpeaks(w, custom_additions, filterSettings =
 getOption("RMassBank")$filterSettings, progressbar = "progressBarHook")
reanalyzeFailpeak(mass, custom_additions, cpdID, mode,
  filterSettings = getOption("RMassBank")$filterSettings)
```

## Arguments

w	A 'msmsWorkspace' with annotated peaks.
custom_additions	The allowed additions, e.g. "N2O".
filterSettings	Settings for filtering data. Refer to <a href="#">analyzeMsMs</a> for settings.
progressbar	The progress bar callback to use. Only needed for specialized applications. Cf. the documentation of <a href="#">progressBarHook</a> for usage.
mass	(Usually recalibrated) m/z value of the peak.
cpdID	Compound ID of this spectrum.
mode	for 'reanalyzeFailpeak', the 'mode' (adduct) of the analyzed spectrum.

## Details

`reanalyzeFailpeaks` examines the `unmatchedPeaksC` table in `specs` and sends every peak through `reanalyzeFailpeak`.

## Value

The aggregate data frame extended by the columns: #'

`reanalyzed.???` If reanalysis (step 7) has already been processed: matching values from the reanalyzed peaks

`matchedReanalysis`

Whether reanalysis has matched (TRUE), not matched(FALSE) or has not been conducted for the peak(NA).

It would be good to merge the analysis functions of `analyzeMsMs` with the one used here, to simplify code changes.

**Author(s)**

Michael Stravs

**See Also**

[analyzeMsMs](#), [msmsWorkflow](#)

**Examples**

```
## As used in the workflow:  
## Not run:  
reanalyzedRcSpecs <- reanalyzeFailpeaks(w@aggregated, custom_additions="N20", mode="pH")  
# A single peak:  
reanalyzeFailpeak("N20", 105.0447, 1234, 1, 1)  
  
## End(Not run)
```

---

recalibrate

*Predefined recalibration functions.*

---

**Description**

Predefined fits to use for recalibration: Loess fit and GAM fit.

**Usage**

```
recalibrate.loess(rcdata)  
  
recalibrate.identity(rcdata)  
  
recalibrate.mean(rcdata)  
  
recalibrate.linear(rcdata)
```

**Arguments**

rcdata	A data frame with at least the columns <code>recalfield</code> and <code>mzFound</code> . <code>recalfield</code> will usually contain delta(ppm) or delta(mz) values and is the target parameter for the recalibration.
--------	--

## Details

`recalibrate.loess()` provides a Loess fit (`recalibrate.loess`) to a given recalibration parameter. If MS and MS/MS data should be fit together, `recalibrate.loess` provides good default settings for Orbitrap instruments.

`recalibrate.identity()` returns a non-recalibration, i.e. a predictor which predicts 0 for all input values. This can be used if the user wants to skip recalibration in the RMassBank workflow.

`#' recalibrate.mean()` and `recalibrate.linear()` are simple recalibrations which return a constant shift or a linear recalibration. They will be only useful in particular cases.

`recalibrate()` itself is only a dummy function and does not do anything.

Alternatively other functions can be defined. Which functions are used for recalibration is specified by the RMassBank options file. (Note: if `recalibrateMS1: common`, the `recalibrator: MS1` value is irrelevant, since for a common curve generated with the function specified in `recalibrator: MS2` will be used.)

## Value

Returns a model for recalibration to be used with `predict` and the like.

## Author(s)

Michael Stravs, EAWAG <[michael.stravs@eawag.ch](mailto:michael.stravs@eawag.ch)>

## Examples

```
## Not run:
rcdata <- subset(spec$peaksMatched, formulaCount==1)
ms1data <- recalibrate.addMS1data(spec, mode, 15)
rcdata <- rbind(rcdata, ms1data)
rcdata$recalfield <- rcdata$dppm
rcCurve <- recalibrate.loess(rcdata)
# define a spectrum and recalibrate it
s <- matrix(c(100,150,200,88.8887,95.0005,222.2223), ncol=2)
colnames(s) <- c("mz", "int")
recalS <- recalibrateSingleSpec(s, rcCurve)

Alternative: define an custom recalibrator function with different parameters
recalibrate.MyOwnLoess <- function(rcdata)
{
  return(loess(recalfield ~ mzFound, data=rcdata, family=c("symmetric"),
    degree = 2, span=0.4))
}
# This can then be specified in the RMassBank settings file:
# recalibrateMS1: common
# recalibrator:
#   MS1: recalibrate.loess
#   MS2: recalibrate.MyOwnLoess")
# [...]

## End(Not run)
```

---

```
recalibrate.addMS1data
```

*Return MS1 peaks to be used for recalibration*

---

## Description

Returns the precursor peaks for all MS1 spectra in the spec dataset with annotated formula to be used in recalibration.

For all spectra in spec\$specFound, the precursor ion is extracted from the MS1 precursor spectrum. All found ions are returned in a data frame with a format matching spec\$peaksMatched and therefore suitable for rbind to the spec\$peaksMatched table. However, only minimal information needed for recalibration is returned.

## Usage

```
recalibrate.addMS1data(spec, recalibrateMS1Window =  
 getOption("RMassBank")$recalibrateMS1Window)
```

## Arguments

spec	A msmsWorkspace or RmbSpectraSetList containing spectra for which MS1 "peaks" should be "constructed".
recalibrateMS1Window	Window width to look for MS1 peaks to recalibrate (in ppm).

## Value

A data frame with columns mzFound, formula, mzCalc, dppm,dbe,int,dppmBest, formulaCount, good, cpdID, scan, parentScan, dppmRc. However, columnsdbe,int, formulaCount, good, scan, parentScan do not contain real information and are provided only as fillers.

## Author(s)

Michael Stravs, EAWAG <michael.stravs@eawag.ch>

## Examples

```
## Not run:  
# More or less as used in recalibrateSpectra:  
rcdata <- peaksMatched(w)  
rcdata <- rcdata[rcdata$formulaCount == 1, ,drop=FALSE]  
ms1data <- recalibrate.addMS1data(w, "pH", 15)  
rcdata <- rbind(rcdata, ms1data)  
# ... continue constructing recalibration curve with rcdata  
  
## End(Not run)
```

---

RmbDefaultSettings      *RMassBank settings*

---

## Description

Load, set and reset settings for RMassBank.

## Usage

```
loadRmbSettings(file_or_list)  
  
loadRmbSettingsFromEnv(env = .GlobalEnv)  
  
RmbDefaultSettings()  
  
RmbSettingsTemplate(target)
```

## Arguments

file_or_list	The file (YML or R format) or R list with the settings to load.
target	The path where the template setting file should be stored.
env	The environment to load the settings from.

## Details

RmbSettingsTemplate creates a template file in which you can adjust the settings as you like. Before using RMassBank, you must then load the settings file using loadRmbSettings. RmbDefaultSettings loads the default settings. loadRmbSettingsFromEnv loads the settings stored in env\$RmbSettings, which is useful when reloading archives with saved settings inside.

Note: no settings are loaded upon loading MassBank! This is intended, so that one never forgets to load the correct settings.

The settings are described in [RmbSettings](#).

## Value

None.

## Note

**The default settings will not work for you unless you have, by chance, installed OpenBabel into the same directory as I have!**

## Author(s)

Michael Stravs

## See Also

[RmbSettings](#)

## Examples

```
# Create a standard settings file and load it (unedited)
RmbSettingsTemplate("mysettings.ini")
loadRmbSettings("mysettings.ini")
unlink("mysettings.ini")
```

---

RmbSettings

*RMassBank settings*

---

## Description

Describes all settings for the RMassBank settings file.

## Details

- **deprofile** Whether and how to deprofile input raw files. Leave the setting empty if your raw files are already in "centroid" mode. If your input files are in profile mode, you have the choice between algorithms `deprofile.spline`, `deprofile.fwhm`, `deprofile.localMax`; refer to the individual manpages for more information.
- **rtMargin**, **rtShift** The allowed retention time deviation relative to the values specified in your compound list (see [loadList](#)), and the systematic shift (due to the use of, e.g., pre-columns or other special equipment).
- **babeldir** Directory to OpenBabel. Required for creating molfiles for MassBank export. If no OpenBabel directory is given, RMassBank will attempt to use the CACTUS webservice for SDF generation. It is strongly advised to install OpenBabel; the CACTUS structures have explicit hydrogen atoms. The path should point to the directory where babel.exe (or the Linux "babel" equivalent) lies.
- **use\_version** Which MassBank record format to use; version 2 is strongly advised, version 1 is considered outdated and should be used only if for some reason you are running old servers and an upgrade is not feasible.
- **use\_rean\_peaks** Whether to include peaks from reanalysis (see [reanalyzeFailpeaks](#)) in the MassBank records. Boolean, TRUE or FALSE.
- **annotations** A list of constant annotations to use in the MassBank records. The entries `authors`, `copyright`, `license`, `instrument`, `instrument_type`, `compound_class` correspond to the MassBank entries AUTHORS, COPYRIGHT, PUBLICATION, LICENSE, AC\$INSTRUMENT, AC\$INSTRUMENT\_TYPE, CH\$COMPOUND\_CLASS. The entry `confidence_comment` is added as COMMENT: CONFIDENCE entry.  
The entry `internal_id_fieldname` is used to name the MassBank entry which will keep a reference to the internal compound ID used in the workflow: for `internal_id_fieldname` = MYID and e.g. compound 1234, an entry will be added to the MassBank record with COMMENT: MYID 1234. The internal fieldname should not be left empty!

The entries `lc_gradient`, `lc_flow`, `lc_solvent_a`, `lc_solvent_b`, `lc_column` correspond to the MassBank entries AC\$CHROMATOGRAPHY: FLOW\_GRADIENT, FLOW\_RATE, SOLVENT A, SOLVENT B, COLUMN\_NAME.

`ms_type`, `ionization` correspond to AC\$MASS\_SPECTROMETRY: MS\_TYPE, IONIZATION.

`entry_prefix` is the two-letter prefix used when building MassBank accession codes.

Entries under `ms_dataprocessing` are added as MS\$DATA\_PROCESSING: entries, in addition to the default WHOLE: RMassBank.

- `annotator` For advanced users: option to select your own custom annotator. Check [annotator.default](#) and the source code for details.
- `spectraList` This setting describes the experimental annotations for the single data-dependent scans. For every data-dependent scan event, a `spectraList` entry with `mode`, `ces`, `ce`, `res` denoting collision mode, collision energy in short and verbose notation, and FT resolution.
- `accessionNumberShifts` This denotes the starting points for accession numbers for different ion types. For example, pH: 0, mH: 50 means that [M+H]+ spectra will start at XX123401 (XX being the `entry_prefix` and 1234 the compound id) and [M-H]- will start at XX123451.
- `electronicNoise`, `electronicNoiseWidth` Known electronic noise peaks and the window to be used by [cleanElnoise](#)
- `recalibrateBy` dppm or dmz to recalibrate either by delta ppm or by delta m/z.
- `recalibrateMS1` common or separate to recalibrate MS1 data points together or separately from MS2 data points.
- `recalibrator: MS1, MS2` The functions to use for recalibration of MS1 and MS2 data points. Note that the MS1 setting is only meaningful if `recalibrateMS1: separate`, otherwise the MS2 setting is used for a common recalibration curve. See [recalibrate.loess](#) for details.
- `multiplicityFilter` Define the multiplicity filtering level. Default is 2, a value of 1 is off (no filtering) and >2 is harsher filtering.
- `titleFormat` The title of MassBank records is a mini-summary of the record, for example "Dinotefuran; LC-ESI-QFT; MS2; CE: 35 By default, the first compound name CH\$NAME, instrument type AC\$INSTRUMENT\_TYPE, MS/MS type AC\$MASS\_SPECTROMETRY: MS\_TYPE, collision energy RECORD\_TITLE\_CE, resolution AC\$MASS\_SPECTROMETRY: RESOLUTION and precursor MS\$FOCUSED\_ION: PRECURSOR\_TYPE are used. If alternative information is relevant to differentiate acquired spectra, the title should be adjusted. For example, many TOFs do not have a resolution setting. See MassBank documentation for more.
- `filterSettings` A list of settings that affect the MS/MS processing. The entries `ppmHighMass`, `ppmLowMass`, `massRangeDivision` set values for pre-processing, prior to recalibration. `ppmHighMass` defines the ppm error for the high mass range (default 10 ppm for Orbitraps), `ppmLowMass` is the error for the low mass range (default 15 ppm for Orbitraps) and `massRangeDivision` is the m/z value defining the split between the high and low mass range (default m/z = 120).  
The entry `ppmFine` defines the ppm cut-off post recalibration. The default value of 5 ppm is recommended for Orbitraps. For other instruments this can be interpreted from the recalibration plot. All ppm limits are one-sided (e.g. this includes values to +5 ppm or -5 ppm deviation from the exact mass).
- The entries `prelimCut`, `prelimCutRatio` define the intensity cut-off and cut-off ratio (in the peak selection for the recalibration only. Careful: the default value 1e4 for Orbitrap LTQ positive mode could remove all peaks for TOF data and will remove too many peaks for Orbitrap LTQ negative mode spectra!

The entry `specOKLimit` defines the intensity limit to include MS/MS spectra. MS/MS spectra must have at least one peak above this limit to proceed through the workflow.

`dbeMinLimit` defines the minimum allowable ring and double bond equivalents (DBE) allowed for assigned formulas. This assumes maximum valences for elements with multiple valence states. The default is -0.5 (accounting for fragments being ions).

The entries `satelliteMzLimit`, `satelliteIntLimit` define the cut-off m/z and intensity values for satellite peak removal (an artefact of Fourier Transform processing). All peaks within the m/z limit (default 0.5) and intensity ratio (default 0.05 or 5 Fourier Transform instruments only (e.g. Orbitrap).

- `filterSettings` Parameters for adjusting the raw data retrieval. The entry `ppmFine` defines the ppm error to look for the precursor in the MS1 (parent) spectrum. Default is 10 ppm for Orbitrap.

`mzCoarse` defines the error to search for the precursor specification in the MS2 spectrum. This is often only saved to 2 decimal places and thus can be quite inaccurate. The accuracy also depends on the isolation window used. The default settings (for e.g. Orbitrap) is 0.5 (Da, or Th for m/z).

The entry `fillPrecursorScan` is largely untested. The default value (FALSE) assumes all necessary precursor information is available in the mzML file. A setting of TRUE tries to fill in the precursor data scan number if it is missing. Only tested on one case study so far - feedback welcome!

## Author(s)

Michael Stravs, Emma Schymanski

## See Also

[loadRmbSettings](#)

---

RmbSpectraSet-class    *Set of spectra pertaining to one compound*

---

## Description

Set of spectra pertaining to one compound

## Slots

`parent` `Spectrum1` The precursor spectrum

`children` `RmbSpectrum2List` List of ‘`RmbSpectrum2`’ objects for the fragmentation spectra, which are first extracted and later processed during ‘`msmsWorkflow`’

`found` logical, denotes whether or not fragmentation spectra were found for this compound

`complete` logical, denotes whether or not \*all\* expected collision energies were found for this compound

`empty` logical, ‘TRUE’ if there are zero found spectra for this compound

`formula` character, the molecular formula of the neutral compound  
`id` The ID of the compound in the RMassBank compound list (see [loadList](#))  
`mz` the m/z value of the precursor  
`name` The name of the compound  
`mode` The ion type of the precursor, e.g. ‘pH, mH, mNa’  
`smiles` the SMILES string for the compound structure

---

RmbSpectrum2-class *RMassBank Representation of an MSMS Spectrum*

---

## Description

This extends the `Spectrum2` class of the `MSnbase` package and introduces further slots that are used to store information during the `RMassBank` workflow.

## Slots

`satellite` logical If TRUE, the corresponding peak was removed as satellite.  
`low` logical If TRUE, the corresponding peak was removed because it failed the intensity cutoff.  
`rawOK` logical If TRUE, the peak passed satellite and low-intensity cutoff removal.  
`good` logical If TRUE, a formula could be found for the peak and the peak passed all filter criteria. (see the `RMassBank` vignette or the documentation of [analyzeMsMs#](#) for details on filter settings)  
`mzCalc` numeric The mz value calculated from the found formula for each peak (if any)  
`formula` character The formula found for each peak. `generate.formula` is used for formula-fitting  
`dbe` numeric The number of double bond equivalents. This is calculated from the found formula for each peak (if any)  
`formulaCount` integer The number of different formulae found for each peak. Note: A peak for which multiple formulas were found will appear multiple times. Hence there may be multiple entries in the `formula`, `dppm` and `mzCalc` slot for the same mz value.  
`formulaSource` character "analyze" or "reanalysis" Shows whether the current formula for the peak was determined by normal analysis ("analyze") or by reanalysis of a failpeak ("reanalysis")  
`dppm` numeric The ppm deviation of the mz value from the found formula (if any).  
`dppmBest` numeric The ppm deviation of the mz value from the best formula found.  
`ok` logical one-element vector If this is TRUE, the spectrum was successfully processed with at least one resulting peak. Otherwise, one of the following cases applies:
 

- All peaks failed the intensity cutoff i.e. the whole spectrum contains low intensity peaks, only.
- All peaks were marked as satellites.

- All peaks in the spectrum have a lower intensity than the value given in the `specOkLimit` filter setting. (see the RMassBank vignette or the documentation of `analyzeMsMs`)
- The precursor ion formula is invalid (see `is.valid.formula`)
- The spectrum is empty.
- No molecular formula could be found for any of the peaks.
- All peaks failed the `dbeMinLimit` criterion. (see the RMassBank vignette or the documentation of `analyzeMsMs`)

`info` list Spectrum identifying information (collision energy, resolution, collision mode) from the `spectralList`

`properties` `data.frame` This is used as a flexible placeholder to store additional properties for each peak throughout the workflow. After the last step of the `mbWorkflow`, this will typically contain columns `mzRaw`, `noise`, `formulaMultiplicity`, `bestMultiplicity` and `filterOK`. However, new columns may be added on demand (see `property<-`)

## See Also

`generate.formula`, `property<-` `analyzeMsMs`, `generate.formula`, `is.valid.formula`

---

### RmbSpectrum2List-class

*SimpleList specializations*

---

## Description

Typed lists using `SimpleList`

---

<code>rmb_log_debug</code>	<i>Pass arguments to <code>logger::log_debug</code> using custom RMassBank-logging settings</i>
----------------------------	---

---

## Description

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

## Usage

`rmb_log_debug(....)`

## Arguments

`...` The log message, as for ‘`logger::log_...`’ functions

## Author(s)

pstahlhofen

**See Also**`logger::log_debug`

---

<code>rmr_log_error</code>	<i>Pass arguments to <code>logger::log_error</code> using custom RMassBank-logging settings</i>
----------------------------	---

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**`rmr_log_error(...)`**Arguments**

... The log message, as for ‘`logger::log_...`’ functions

**Author(s)**

pstahlhofen

**See Also**`logger::log_error`

---

<code>rmr_log_fatal</code>	<i>Pass arguments to <code>logger::log_fatal</code> using custom RMassBank-logging settings</i>
----------------------------	---

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**`rmr_log_fatal(...)`**Arguments**

... The log message, as for ‘`logger::log_...`’ functions

**Author(s)**

pstahlhofen

**See Also**

logger::log\_fatal

---

rmb_log_info	<i>Pass arguments to logger::log_info using custom RMassBank-logging settings</i>
--------------	---

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**

`rmb_log_info(...)`

**Arguments**

... The log message, as for ‘logger::log\_...’ functions

**Author(s)**

pstahlhofen

**See Also**

logger::log\_info

---

rmb_log_success	<i>Pass arguments to logger::log_success using custom RMassBank-logging settings</i>
-----------------	--

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**

`rmb_log_success(...)`

**Arguments**

... The log message, as for ‘logger::log\_...’ functions

**Author(s)**

pstahlhofen

**See Also**

`logger::log_success`

---

<code>rmb_log_trace</code>	<i>Pass arguments to <code>logger::log_trace</code> using custom RMassBank-logging settings</i>
----------------------------	---

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**

`rmb_log_trace(...)`

**Arguments**

`...` The log message, as for ‘`logger::log_...`’ functions

**Author(s)**

`pstahlhofen`

**See Also**

`logger::log_trace`

---

<code>rmb_log_warn</code>	<i>Pass arguments to <code>logger::log_warn</code> using custom RMassBank-logging settings</i>
---------------------------	--

---

**Description**

The logging file to be used can be specified by the user in the `logging_file` field of `settings.ini`

**Usage**

`rmb_log_warn(...)`

**Arguments**

`...` The log message, as for ‘`logger::log_...`’ functions

**Author(s)**

`pstahlhofen`

**See Also**`logger::log_warn`

---

<code>selectPeaks</code>	<i>Select peaks from aggregate table</i>
--------------------------	--

---

**Description**

Selects peaks from aggregate table according to different criteria.

**Usage**

```
selectPeaks(o, ...)

## S4 method for signature 'RmbSpectrum2'
selectPeaks(o, filter, ..., enclos = parent.frame(2))

## S4 method for signature 'Spectrum'
selectPeaks(o, filter, ..., enclos = parent.frame(2))

## S4 method for signature 'RmbSpectrum2List'
selectPeaks(o, ..., enclos = parent.frame(2))

## S4 method for signature 'RmbSpectraSetList'
selectPeaks(o, ..., enclos = parent.frame(2))

## S4 method for signature 'RmbSpectraSet'
selectPeaks(o, ..., enclos = parent.frame(2))

## S4 method for signature 'data.frame'
selectPeaks(o, filter, ..., enclos = parent.frame(2))

## S4 method for signature 'msmsWorkspace'
selectPeaks(o, ..., enclos = parent.frame(2))
```

**Arguments**

- `o` msmsWorkspace or aggregate data.frame from a workspace.
- `...` no additional parameters
- `filter` The expression (to be evaluated in context of the ‘`getData()`‘ result on the spectrum) to define the peaks to keep. For example, ‘`good & filterOK`‘
- `enclos` The context in which to evaluate the filter expression, by default set such that the spectrum ‘`getData()`‘ is retrieved.

**Value**

Peak dataframe according to the specified criteria.

**Methods (by class)**

- `selectPeaks(RmbSpectraSetList)`: A method to filter spectra to the specified peaks
- `selectPeaks(RmbSpectraSet)`: A method to filter spectra to the specified peaks
- `selectPeaks(data.frame)`: A method to retrieve the specified peaks from the "aggregated" slot (a data.frame object) in an msmsWorkSpace
- `selectPeaks(msmsWorkspace)`: A method to retrieve the specified peaks from an msmsWorkSpace

**Author(s)**

stravsmi

---

<code>selectSpectra</code>	<i>Select a subset of spectra matching properties</i>
----------------------------	---

---

**Description**

From a list of `RmbSpectraSets`, returns the spectra which match a criterion (found, complete, empty as in [checkSpectra](#)). This can be returned either as a TRUE/FALSE vector, as a vector of indices for matching elements, as a vector of `RmbSpectraSet` objects matching the conditions, or as a vector of `RmbSpectraSet` objects NOT matching the conditions (sic!).

**Usage**

```
selectSpectra(s, property, value = "logical")

## S4 method for signature 'RmbSpectraSetList,character'
selectSpectra(s, property, value = "logical")

## S4 method for signature 'msmsWorkspace,character'
selectSpectra(s, property, value = "logical")
```

**Arguments**

<code>s</code>	The <code>RmbSpectraSetList</code> or <code>msmsWorkspace</code> to select <code>RmbSpectraSets</code> from.
<code>property</code>	The property to check (found, complete or empty)
<code>value</code>	<code>logical</code> if a TRUE/FALSE list should be returned; <code>index</code> if a vector of matching indices should be returned, <code>object</code> if matching objects should be returned, <code>mismatch</code> if mismatching objects should be returned.

**Value**

As described above.

## Methods (by class)

- `selectSpectra(s = RmbSpectraSetList, property = character)`: A method for selecting spectra from a spectra set list
- `selectSpectra(s = msmsWorkspace, property = character)`: A method for selecting spectra from an msmsWorkspace

## Author(s)

stravsmi

`setAccessionBuilder`    *Define a programmatic or gluey ACCESSION builder*

## Description

Define a programmatic or gluey ACCESSION builder

## Usage

```
setAccessionBuilder(accessionBuilder)
```

## Arguments

accessionBuilder

a function that takes parameters ‘cpd’ (an instance of ‘RmbSpectraSet’), ‘spectrum’ (an instance of ‘RmbSpectrum2’) and ‘subscan’ (an integer denoting relative scan id) and returns a ‘character’. Alternatively a glue string just like the one in the RMassBank settings.

`setData` *Set RmbSpectrum2 data from data.frame*

## Description

Sets all slots which are present as columns in the given data frame. Optionally cleans the object, i.e. empties slots not defined in the data frame.

## Usage

```
## S4 method for signature 'RmbSpectrum2,data.frame'  
setData(s, df, clean = TRUE)
```

**Arguments**

s	The RmbSpectrum2 object to modify
df	The data frame with new data
clean	TRUE if slots which aren't present as columns in the data frame should be cleared.

**Value**

The modified RmbSpectrum2.

**Author(s)**

stravsmi

---

smiles2mass	<i>Calculate the mass from a SMILES-String</i>
-------------	--

---

**Description**

Uses a SMILES-String to calculate the mass using rcdk-integrated functions.

**Usage**

smiles2mass(SMILES)

**Arguments**

SMILES	A String-object representing a SMILES
--------	---------------------------------------

**Value**

The calculated mass of the given SMILES-Formula

**Author(s)**

Erik Mueller

**Examples**

```
## Not run:
smiles2mass("CC(=O)NC(C(0)1)C(0)C(OC(02)C(0)C(OC(03)C(0)C(0)C(0)C(0)3)C(0)C(CO)2)C(CO)01")
## End(Not run)
```

---

spectraCount	<i>Count MS2 spectra per compound</i>
--------------	---------------------------------------

---

## Description

Counts the number of acquired spectra for a compound or multiple compounds

## Usage

```
spectraCount(s)

## S4 method for signature 'RmbSpectraSet'
spectraCount(s)

## S4 method for signature 'RmbSpectraSetList'
spectraCount(s)

## S4 method for signature 'msmsWorkspace'
spectraCount(s)
```

## Arguments

`s` The object (`RmbSpectraSet`, `RmbSpectraSetList` or `msmsWorkspace`) to count the spectra in.

## Value

For `RmbSpectraSet` objects, a single number counting the spectra in that object. For `RmbSpectraSetList` or `msmsWorkspace`, a vector with spectra counts for all compounds (`RmbSpectraSets`) in the object.

## Methods (by class)

- `spectraCount(RmbSpectraSet)`: Counts the number of acquired spectra for an `RmbSpectraSet`
- `spectraCount(RmbSpectraSetList)`: Counts the number of acquired spectra for an `RmbSpectraSetList`
- `spectraCount(msmsWorkspace)`: Counts the number of acquired spectra for an `msmsWorkSpace`

## Author(s)

stravsmi

---

`to.limits.rcdk` *Convert formula to Rcdk limits*

---

## Description

Converts a molecular formula e.g. C15H20 into an upper limit appropriate for use with Rcdk's [generate.formula](#) function's `element` argument.

## Usage

```
to.limits.rcdk(formula)
```

## Arguments

`formula` A molecular formula in string or list representation ("C6H6" or `list(C=6,H=6)`).

## Details

This helper function is used to make the upper limits for [generate.formula](#) when finding subformulas to match to a MS2 fragment peak.

## Value

An array in the form `c( c("C", "0", "12"), c("H", "0", "12"))` (for input of "C12H12").

## Author(s)

Michael Stravs

## See Also

[generate.formula](#), [add.formula](#)

## Examples

```
#  
to.limits.rcdk("C6H6")  
to.limits.rcdk(add.formula("C6H1206", "H"))
```

---

**toMassbank***Write MassBank record into character array*

---

## Description

Writes a MassBank record in list format to a text array.

## Usage

```
toMassbank(o, ...)

## S4 method for signature 'RmbSpectraSet'
toMassbank(o, addAnnotation = getOption("RMassBank")$add_annotation)

## S4 method for signature 'RmbSpectrum2'
toMassbank(o, addAnnotation = getOption("RMassBank")$add_annotation)
```

## Arguments

- o An object to convert to MassBank record format. This may be a single 'RmbSpectrum2', or a complete compound (an 'RmbSpectraSet'),
- ... Parameters passed to the implementation, in particular 'addAnnotation'
- addAnnotation 'logical', whether to add peak annotations (putative formulas) to the record.

## Details

The function is a general conversion tool for the MassBank format; i.e. the field names are not fixed. `mbdata` must be a named list, and the entries can be as follows:

- A single text line:  
`'CH\\$EXACT_MASS' = '329.1023'`  
is written as  
`CH\\$EXACT_MASS: 329.1023`
- A character array:  
`'CH\\$NAME' = c('2-Aminobenzimidazole', '1H-Benzimidazol-2-amine')`  
is written as  
`CH\\$NAME: 2-Aminobenzimidazole`  
`CH\\$NAME: 1H-Benzimidazol-2-amine`
- A named list of strings:  
`'CH\\$LINK' = list('CHEBI' = "27822", "KEGG" = "C10901")`  
is written as  
`CH\\$LINK: CHEBI 27822`  
`CH\\$LINK: KEGG C10901`
- A data frame (e.g. the peak table) is written as specified in the MassBank record format (Section 2.6.3): the column names are used as headers for the first line, all data rows are printed space-separated.

**Value**

The result is a text array, which is ready to be written to the disk as a file.

**Note**

The function iterates over the list item names. **This means that duplicate entries in mbdata are (partially) discarded!** The correct way to add them is by making a character array (as specified above): Instead of 'CH\\$NAME' = 'bla', 'CH\\$NAME' = 'blub' specify 'CH\\$NAME' = c('bla', 'blub').

**Author(s)**

Michael Stravs

**References**

MassBank record format: [http://www.massbank.jp/manuals/MassBankRecord\\_en.pdf](http://www.massbank.jp/manuals/MassBankRecord_en.pdf)

**See Also**

[buildRecord](#), [mbWorkflow](#)

**Examples**

```
## Not run:
# Read just the compound info skeleton from the Internet for some compound ID
id <- 35
mbdata <- gatherData(id)
#' # Export the mbdata blocks to line arrays
# (there is no spectrum information, just the compound info...)
mbtext <- toMassbank(mbdata)

## End(Not run)
```

**Description**

Converts a pseudospectrum extracted from XCMS using CAMERA into the msmsWorkspace(at)spectrum-format that RMassBank uses

**Usage**

`toRMB(msmsXCMSSpecs, cpdID, mode, MS1spec)`

**Arguments**

msmsXCMSSpecs	The compoundID of the compound that has been used for the peaklist
cpdID	The compound ID of the substance of the given spectrum
mode	The ionization mode that has been used for the spectrum
MS1spec	The MS1-spectrum from XCMS, which can be optionally supplied

**Value**

One list element of the (at)specs-entry from an msmsWorkspace

**Author(s)**

Erik Mueller

**See Also**

[msmsWorkspace-class](#)

**Examples**

```
## Not run:  
XCMSPspectra <- findmsmsHRperxcms.direct("Glucolesquerellin_2184_1.mzdata", 2184)  
wspecs <- toRMB(XCMSPspectra)  
  
## End(Not run)
```

---

updateHeader                    *Add a header to a Multiblock JCAMP file*

---

**Description**

JCAMP files containing multiple blocks are usually structured by so-called link blocks. If no link block is present, the readJDX package is not able to parse the file. This method will add a link block at the top of the given file or print a message if an existing link block is found. The file is not changed in this case.

**Usage**

```
updateHeader(filename)
```

**Arguments**

filename	character The name of the file to which a link block should be added. The filename is also used as content for the TITLE field in the link block
----------	--

**Value**

Nothing is returned

**Author(s)**

pstahlhofen

**Examples**

```
## Not run:
updateHeader("my_multiblock_jcamp.jdx")

## End(Not run)
```

updateSettings

*Update settings to current version***Description**

Checks if all necessary fields are present in the current settings and fills in default values from the [RmbDefaultSettings](#) if required.

**Usage**

```
updateSettings(settings, warn = TRUE)
```

**Arguments**

settings	The set of settings to check and update.
warn	Whether to update parameters quietly (FALSE) or to notify the user of the changed parameters (TRUE, default.) This serves to make the user aware that standard parameters are filled in!

**Value**

The updated set of settings.

**Note**

Important: There is a change in behaviour of RMassBank in certain cases when `filterSettings` is not present in the old settings! The default pre-recalibration cutoff from [RmbDefaultSettings](#) is 10000. Formerly the pre-recalibration cutoff was set to be 10000 for positive spectra but 0 for negative spectra.

Updating the settings files is preferred to using the `updateSettings` function.

**Author(s)**

Stravs MA, Eawag <michael.stravs@eawag.ch>

## Examples

```
## Not run:  
w@settings <- updateSettings(w@settings)  
  
## End(Not run)
```

---

validate

*Validate MassBank records with a set of Unit tests*

---

## Description

Validates a plain text MassBank record, or recursively all records within a directory. The Unit Tests to be used are installed in RMassBank/inst/validationTests and currently include checks for NAs, peaks versus precursor, precursor mz, precursor type, SMILES vs exact mass, total intensities and title versus type. The validation report is saved as "report.html" in the working directory.

## Usage

```
validate(path, simple = TRUE)
```

## Arguments

path	The filepath to a single record, or a directory to search recursively
simple	If TRUE the function creates a simpler form of the RUnit .html report, better readable for humans. If FALSE it returns the unchanged RUnit report.

## Examples

```
## Not run:  
validate("/tmp/MassBank/OpenData/record/")  
  
## End(Not run)
```

# Index

+ ,RmbSpectraSet,ANY-method, 5  
+ ,RmbSpectrum2List,ANY-method, 5  
+ ,Spectrum,numeric-method, 6  
- ,RmbSpectraSet,ANY-method, 6  
- ,RmbSpectrum2List,ANY-method, 7  
- ,Spectrum,numeric-method, 7  
.RmbSpectraSet (RmbSpectraSet-class),  
    109  
.RmbSpectraSetList  
    (RmbSpectrum2List-class), 111  
.RmbSpectrum2 (RmbSpectrum2-class), 110  
.RmbSpectrum2List  
    (RmbSpectrum2List-class), 111  
.msmsWorkspace (msmsWorkspace-class), 88  
.parseTitleString, 8  
.updateObject.RmbSpectrum2.formulaSource,  
    9  
  
add.formula, 9, 55, 73, 91, 120  
addMB, 10  
addPeaks, 11, 20, 81  
addPeaksManually, 11, 12  
addProperty, 13  
addProperty,data.frame,character,character-method  
    (addProperty), 13  
addProperty,RmbSpectrum2,character,character-method  
    (addProperty), 13  
aggregateSpectra, 14, 24  
analyzeMsMs, 15, 15, 19, 20, 36, 38–40, 87,  
    102, 103, 110, 111  
annotator.default, 18, 108  
archiveResults, 19, 90  
  
buildRecord, 8, 19, 122  
buildRecord,RmbSpectraSet-method  
    (buildRecord), 19  
buildRecord,RmbSpectrum2-method  
    (buildRecord), 19  
  
CAS2SMILES, 21  
  
checkIsotopes, 21  
checkSpectra, 23, 116  
checkSpectra,RmbSpectraSet,character-method  
    (checkSpectra), 23  
cleanElnoise, 24, 108  
cleanElnoise,data.frame,numeric,numeric-method  
    (cleanElnoise), 24  
cleanElnoise,RmbSpectraSet,numeric,numeric-method  
    (cleanElnoise), 24  
cleanElnoise,RmbSpectrum2,numeric,numeric-method  
    (cleanElnoise), 24  
cleanElnoise,RmbSpectrum2List,numeric,numeric-method  
    (cleanElnoise), 24  
combineMultiplicities, 25  
compoundlist2SDF, 26  
createCompoundlist, 27  
createMolfile, 28, 33, 34  
CTS.externalIdSubset, 29  
CTS.externalIdTypes, 30  
  
dbe, 30  
deprofile, 31, 44, 47, 107  
  
exportMassbank, 33  
fillback, 34  
fillback,msmsWorkspace,missing,missing-method  
    (fillback), 34  
fillback,RmbSpectraSet,missing,data.frame-method  
    (fillback), 34  
fillback,RmbSpectrum2,character,data.frame-method  
    (fillback), 34  
filterCompoundlist, 35  
filterLowaccResults, 18, 36, 39  
filterMultiplicity, 37  
filterPeakSatellites, 16–18, 36, 38  
filterPeaksMultiplicity, 37, 38, 39  
findCAS (findMz), 50  
findEIC, 40  
findFormula (findMz), 50

findLevel (findMz), 50  
findMass, 42, 51  
findMsMsHR, 16, 43, 46, 48, 76, 78  
findMsMsHR.direct, 46  
findMsMsHR.ticMS2 (findMsMsHR.ticms2), 47  
findMsMsHR.ticms2, 47  
findMsMsHRperMsp, 48  
findMsMsHRperxcms, 49  
findMz, 42, 50, 52, 75  
findMz.formula, 51, 52  
findName (findMz), 50  
findProgress, 53  
findRt (findMz), 50  
findSmiles, 28  
findSmiles (findMz), 50  
flatten, 53  
formulastring.to.list, 10, 52, 55

gatherCCTE, 56  
gatherData, 53, 54, 57  
gatherDataBabel, 58  
gatherDataUnknown, 59  
gatherPubChem, 60  
generate.formula, 110, 111, 120  
getAnalyticalInfo, 61  
getCactus, 61, 71  
getCASRN, 62  
getCSID, 63  
getCtsKey, 64  
getCtsRecord, 29, 30, 62, 65, 71  
getData, 66, 82, 83  
getData, RmbSpectrum2-method (getData), 66  
getDTXCID, 66  
getDTXSID, 67  
getDTXSMILES, 68  
getField, 69  
getMolecule, 70  
getPcId, 62, 71  
getPrefName, 72

infolist, 53  
is.valid.formula, 10, 55, 73, 91, 111

list.to.formula, 73, 91  
list.to.formula  
    (formulastring.to.list), 55

loadInfolist, 19, 54

loadInfolist (loadInfolists), 74  
loadInfolists, 74, 81  
loadList, 43, 44, 51, 75, 107, 110  
loadMsmsWorkspace (newMsmsWorkspace), 89  
loadRmbSettings, 16, 22, 84, 85, 87, 109  
loadRmbSettings (RmbDefaultSettings), 106  
loadRmbSettingsFromEnv  
    (RmbDefaultSettings), 106

makeMollist, 76  
makePeaksCache, 76  
makeRecalibration, 77  
mbWorkflow, 11, 20, 34, 37, 42, 56–58, 60, 61, 74, 79, 81, 89, 122  
mbWorkspace-class, 80  
mergePeaks, 81  
mergePeaks, data.frame-method  
    (mergePeaks), 81  
mergePeaks, matrix-method (mergePeaks), 81  
mergePeaks, RmbSpectrum2-method  
    (mergePeaks), 81  
mergePeaks, Spectrum-method  
    (mergePeaks), 81  
mergeSpectra, 82  
mergeSpectra, RmbSpectrum2List-method  
    (mergeSpectra), 82  
msmsRead, 83, 87  
msmsRead.RAW, 85  
msmsWorkflow, 12, 15, 18, 25, 26, 50, 79, 85, 86, 88, 90, 98, 103  
msmsWorkspace-class, 88  
multiply.formula (add.formula), 9

newMbWorkspace, 89  
newMsmsWorkspace, 89  
normalize, 20  
normalize, RmbSpectrum2-method, 90  
normalize, RmbSpectrum2List-method, 91

order.formula, 10, 55, 73, 91

parse.smiles, 70  
parseMassBank, 92  
parseMbRecord, 93  
peaksMatched, 94  
peaksMatched, data.frame-method  
    (peaksMatched), 94

peaksMatched, msmsWorkspace-method  
     (peaksMatched), 94  
 peaksUnmatched, 94  
 peaksUnmatched, data.frame-method  
     (peaksUnmatched), 94  
 peaksUnmatched, msmsWorkspace-method  
     (peaksUnmatched), 94  
 plotMbWorkspaces, 95  
 plotRecalibration, 96  
 ppm, 82, 97  
 problematicPeaks, 37, 38, 98  
 processProblematicPeaks, 99  
 progressBarHook, 84, 85, 87, 99, 102  
 property, 100  
 property, RmbSpectrum2, character-method  
     (property), 100  
 property<-, 101  
 property<-, RmbSpectrum2, character, logical, character-method  
     (property<-), 101  
 property<-, RmbSpectrum2, character, logical, missing-method  
     (property<-), 101  
 property<-, RmbSpectrum2, character, missing, character-method  
     (property<-), 101  
 property<-, RmbSpectrum2, character, missing, missing-method  
     (property<-), 101  
 readMbdata, 19  
 readMbdata (flatten), 53  
 reanalyzeFailpeak, 40  
 reanalyzeFailpeak (reanalyzeFailpeaks), 102  
 reanalyzeFailpeaks, 18, 102, 107  
 recalibrate, 77, 78, 103  
 recalibrate.addMS1data, 105  
 recalibrate.loess, 108  
 recalibrateSingleSpec  
     (makeRecalibration), 77  
 recalibrateSpectra (makeRecalibration), 77  
 resetInfolists (loadInfolists), 74  
 resetList (loadList), 75  
 rmb\_log\_debug, 111  
 rmb\_log\_error, 112  
 rmb\_log\_fatal, 112  
 rmb\_log\_info, 113  
 rmb\_log\_success, 113  
 rmb\_log\_trace, 114  
 rmb\_log\_warn, 114  
 RmbDefaultSettings, 106, 124

RmbSettings, 106, 107, 107  
 RmbSettingsTemplate  
     (RmbDefaultSettings), 106  
 RmbSpectraSet-class, 109  
 RmbSpectraSetList  
     (RmbSpectrum2List-class), 111  
 RmbSpectraSetList-class  
     (RmbSpectrum2List-class), 111  
 RmbSpectrum2-class, 110  
 RmbSpectrum2List-class, 111  
 selectPeaks, 20, 115  
 selectPeaks, data.frame-method  
     (selectPeaks), 115  
 selectPeaks, msmsWorkspace-method  
     (selectPeaks), 115  
 selectPeaks, RmbSpectraSet-method  
     (selectPeaks), 115  
 selectPeaks, RmbSpectraSetList-method  
     (selectPeaks), 115  
 selectPeaks, RmbSpectrum2-method  
     (selectPeaks), 115  
 selectPeaks, RmbSpectrum2List-method  
     (selectPeaks), 115  
 selectPeaks, Spectrum-method  
     (selectPeaks), 115  
 selectSpectra, 116  
 selectSpectra, msmsWorkspace, character-method  
     (selectSpectra), 116  
 selectSpectra, RmbSpectraSetList, character-method  
     (selectSpectra), 116  
 setAccessionBuilder, 117  
 setData, 82, 83, 117  
 setData, RmbSpectrum2, data.frame-method  
     (setData), 117  
 show, mbWorkspace-method  
     (mbWorkspace-class), 80  
 show, msmsWorkspace-method  
     (msmsWorkspace-class), 88  
 smiles2mass, 118  
 spectraCount, 119  
 spectraCount, msmsWorkspace-method  
     (spectraCount), 119  
 spectraCount, RmbSpectraSet-method  
     (spectraCount), 119  
 spectraCount, RmbSpectraSetList-method  
     (spectraCount), 119  
 to.limits.rcdk, 120

toMassbank, [20](#), [34](#), [121](#)  
toMassbank, RmbSpectraSet-method  
    (toMassbank), [121](#)  
toMassbank, RmbSpectrum2-method  
    (toMassbank), [121](#)  
toRMB, [50](#), [122](#)  
  
updateHeader, [123](#)  
updateSettings, [124](#)  
  
validate, [92](#), [93](#), [125](#)