

# Package ‘tofsims’

April 12, 2022

**Type** Package

**Title** Import, process and analysis of Time-of-Flight Secondary Ion  
Mass Spectrometry (ToF-SIMS) imaging data

**Version** 1.22.0

**Author** Lorenz Gerber, Viet Mai Hoang

**Maintainer** Lorenz Gerber <genfys@gmail.com>

**Depends** R (>= 3.3.0), methods, utils, ProtGenerics

**Description** This packages offers a pipeline for import, processing and analysis of ToF-SIMS 2D image data. Import of Iontof and Ulvac-Phi raw or preprocessed data is supported. For rawdata, mass calibration, peak picking and peak integration exist. General functionality includes data binning, scaling, image subsetting and visualization. A range of multivariate tools common in the ToF-SIMS community are implemented (PCA, MCR, MAF, MNF). An interface to the bioconductor image processing package EBImage offers image segmentation functionality.

**License** GPL-3

**Imports** Rcpp (>= 0.11.2), ALS, alsace, signal, KernSmooth, graphics,  
grDevices, stats

**Suggests** EBImage, knitr, rmarkdown, testthat, tofsimsData,  
BiocParallel, RColorBrewer

**Enhances** parallel

**LinkingTo** Rcpp, RcppArmadillo

**VignetteBuilder** knitr

**biocViews** ImmunoOncology, Infrastructure, DataImport,  
MassSpectrometry, ImagingMassSpectrometry, Proteomics,  
Metabolomics

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**NeedsCompilation** yes

**URL** <https://github.com/lorenzgerber/tofsims>

**BugReports** <https://github.com/lorenzgerber/tofsims/issues>

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

**git\_branch** RELEASE\_3\_14

**git\_last\_commit** 6cfbd33

**git\_last\_commit\_date** 2021-10-26

**Date/Publication** 2022-04-12

## R topics documented:

tofsims-package . . . . .	4
addFixedWidth . . . . .	5
addPeaks . . . . .	6
analysis . . . . .	7
analysisName . . . . .	8
baseObject . . . . .	9
baseObject,PrComp-method . . . . .	9
baseObject,PrinComp-method . . . . .	10
binning . . . . .	10
bwApply . . . . .	11
calibPointNew . . . . .	12
calibPoints . . . . .	13
calibration . . . . .	14
changePeakWidth . . . . .	15
check.extension . . . . .	16
computeMNF . . . . .	17
computeNoise . . . . .	18
coordToPixel . . . . .	18
coordToPixel,MassImage,numeric-method . . . . .	19
covDiffCalc . . . . .	19
cReadRawPhi . . . . .	20
ctable . . . . .	21
dim,MassImage-method . . . . .	21
dim,MassSpectra-method . . . . .	22
EigenDecompose . . . . .	22
extract.header.data . . . . .	23
findClosestMatch . . . . .	23
findPeakWidth . . . . .	24
getTOFs . . . . .	25
image . . . . .	26
imageMatrix . . . . .	27
import . . . . .	28
import.raw . . . . .	28
instrument . . . . .	29
iters . . . . .	30
iters,MCR-method . . . . .	31
itzipName . . . . .	31
itzipName<- . . . . .	32

LapackGenEigen	32
legend.col	33
look.for.itzip.property	33
MAF	34
makeTIC	35
makeTIC,MassSpectra-method	35
manualSelectPeaks	36
MassImage	36
MassSpectra	38
MCR-class	39
MNF	40
mz,MassSpectra-method	41
nComp	42
ndim	43
ndim,MassSpectra-method	43
nearestNeighbourMean	44
nnMean	44
nnMNF	45
noPlottingData	46
noPlottingData,PCA-method	46
nPeaks	47
nz	47
overlayPlot	48
parIndicesSearch	49
PCA-class	50
pcaLoadings	51
pcaMAF	52
PCAnalysis	52
pcaScores	53
peakIDs	54
PeakList	55
peakMzs	56
peakPick	57
peaks2Spectra	58
peakWidths	59
plot	60
plot,MassImage,missing-method	61
plot,PeakList,missing-method	62
points	63
poissonScaling	64
PrComp-class	65
PrinComp-class	66
readBIF	67
recalibrate	68
reduceSpectrumResolution	68
removePeaks	69
resids	71
resids,MCR-method	71

RSS	72
RSS,MCR-method	72
scale	73
show,MassImage-method	74
show,MassSpectra-method	74
show,PeakList-method	75
smootherGolay	75
smootherSpline	76
smoothScatter	77
SNR	79
subset	79
unitMassPeaks	80
validMassImageObject	81
validMassSpectraObject	82
validPCAObject	82
validPeakListObject	83
xdim	83
xdim,MassImage-method	84
xdim,PCA-method	84
xdim<-	85
xy	85
xySpec	86
ydim	87
ydim,MassImage-method	87
ydim,PCA-method	88
ydim<-	88
zdim	89
zdim,MassSpectra-method	89
<b>Index</b>	<b>90</b>

---

tofsims-package	<i>ToF-SIMS Toolbox (tofsims)</i>
-----------------	-----------------------------------

---

## Description

ToF-SIMS Toolbox

## Details

Package:	tofsims
Type:	Package
Version:	0.99.2
Date:	15-01-2016
License:	GPL-3
LazyLoad:	yes

Toolbox for Time-of-Flight Secondary Ion Mass-Spectrometry (ToF-SIMS) data processing and analysis. The package facilitates importing of raw data files, loading preprocessed data and a range of multivariate analysis methods that are most commonly applied in the ToF-SIMS community.

### Author(s)

Lorenz Gerber <lorenz.gerber@slu.se>

Viet Mai Hoang <hviet.0906@gmail.com>

---

addFixedWidth                    *Generic method to add/update peak width*

---

### Description

This method will update current upper/lower width for all peaks

### Usage

```
addFixedWidth(object, lowerWidth, upperWidth)
```

```
## S4 method for signature 'PeakList,numeric,numeric'  
addFixedWidth(object, lowerWidth, upperWidth)
```

### Arguments

object	PeakList object
lowerWidth	numeric
upperWidth	numeric

### Value

object PeakList with updated/new peak widths

### Examples

```
library(tofsimsData)  
data(tofsimsData)  
testSpectra<-reduceSpectrumResolution(object = testSpectra, everyN = 4, mode = 'keep')  
testSpectra<-smootherSpline(testSpectra, stepsize = 10, spar = 0.3)  
testSpectra<-smootherGolay(testSpectra, p = 3, n = 5)  
testSpectra<-peakPick(testSpectra, span = 100)  
testSpectra<-addFixedWidth(testSpectra, 0.2, 0.2)  
plot(testSpectra, , mzRange=c(38.5,40.5), type = 'l')
```

---

addPeaks	<i>generic method to add peaks</i>
----------	------------------------------------

---

### Description

This method will allow user to plot and add peaks manually. This method will take all parameters of PeakList plot method.

### Usage

```
addPeaks(object, mzs, width, ...)  
  
## S4 method for signature 'PeakList,missing,numeric'  
addPeaks(object, mzs, width, ...)  
  
## S4 method for signature 'PeakList,numeric,numeric'  
addPeaks(object, mzs, width, ...)
```

### Arguments

object	PeakList object
mzs	numeric vector M/z's where peaks shall be added
width	fixed value to add (m/z)
...	further args

### Value

object updated PeakList object

### Examples

```
library(tofsimsData)  
data(tofsimsData)  
testPeakList<-PeakList(analysisName = analysisName(testSpectra),  
instrument = instrument(testSpectra),  
nz = nz(testSpectra),  
calibration = calibration(testSpectra),  
calibPoints = calibPoints(testSpectra),  
mz = mz(testSpectra),  
peakIDs = NULL,  
peakMzs = NULL)  
par(mfcol=c(1,2))  
plot(testPeakList, mzRange=c(25,32), type = 'l')  
testPeakList<-addPeaks(testPeakList, mzs=26:31, width=0.4)  
plot(testPeakList, mzRange=c(25,32), type = 'l')
```

---

analysis	analysis, <i>slot of MassSpectra class objects</i>
----------	--

---

### Description

analysis, slot of MassSpectra class objects

### Usage

```
analysis(object, noAccess, ...)  
  
analysis(object) <- value  
  
## S4 method for signature 'MassSpectra,missing'  
analysis(object)  
  
## S4 method for signature 'MassSpectra,numeric'  
analysis(object, noAccess)  
  
## S4 replacement method for signature 'MassSpectra'  
analysis(object) <- value
```

### Arguments

object	object of class MassSpectra
noAccess	numeric access number to analysis slot
...	additional args
value	object to be put in analysis slot

### Value

summary or content of analysis slot

### See Also

object [MassSpectra](#) other slots [mz](#) [nz](#) [analysisName](#) [instrument](#) [calibPoints](#) [calibration](#)

### Examples

```
library(tofsimsData)  
data(tofsimsData)  
testImage<-PCAnalysis(testImage, nComp = 3)  
## obtain summary of analysis slot content  
analysis(testImage)
```

---

analysisName	analysisName, <i>slot of MassSpectra class objects</i>
--------------	--

---

### Description

analysisName, slot of MassSpectra class objects

### Usage

```
analysisName(object, ...)  
  
analysisName(object) <- value  
  
## S4 method for signature 'MassSpectra'  
analysisName(object)  
  
## S4 replacement method for signature 'MassSpectra'  
analysisName(object) <- value
```

### Arguments

object	object of class MassSpectra
...	further args
value	character replacement value for slot analysisName

### Value

content of analysisName slot

### See Also

object [MassSpectra](#) other slots [mz](#) [analysis](#) [nz](#) [instrument](#) [calibPoints](#) [calibration](#)

### Examples

```
library(tofsimsData)  
data(tofsimsData)  
## access name of analysis  
analysisName(testSpectra)  
## replace name of analysis  
analysisName(testSpectra) <- 'sample001_pos001_settings_default'  
analysisName(testSpectra)
```



---

baseObject                      *generic accessor method baseObject*

---

**Description**

generic accessor method baseObject

**Usage**

baseObject(object)

**Arguments**

object                      helper for prcomp and princomp wrappers

**Value**

baseObject

---

baseObject,PrComp-method  
*constructor for PrComp*

---

**Description**

constructor for PrComp

**Usage**

```
## S4 method for signature 'PrComp'  
baseObject(object)
```

**Arguments**

object                      object of class

**Value**

object of class PrComp

---

baseObject,PrinComp-method  
*constructor for PrinComp*

---

**Description**

constructor for PrinComp

**Usage**

```
## S4 method for signature 'PrinComp'
baseObject(object)
```

**Arguments**

object            object with class

**Value**

object of class PrinComp

---

binning            *binning*

---

**Description**

binning

**Usage**

```
binning(object, binningFactor, ...)

## S4 method for signature 'MassImage'
binning(object, binningFactor = 2)
```

**Arguments**

object            object of class MassImage  
binningFactor    numeric factor for binning (2, 4, etc)  
...                additional args

**Details**

binning is used to reduce the resolution/size of MassImage objects. Optionally mclapply from the parallel package is used to speed up processing time.

**Value**

binned object of class MassImage

**Examples**

```
library(BiocParallel)
testImage<-MassImage('dummy')
par(mfcol=c(1,2), oma=c(0,0,0,0), mar=c(0,0,0,0))
image(testImage)
## the following param will cause to run non parallel
register(SerialParam(), default=TRUE)
testImage <- binning(testImage,binningFactor = 4)
image(testImage)
## Not run:
library(tofsimsData)
data(tofsimsData)
par(mfcol=c(1,2), oma=c(0,0,0,0), mar=c(0,0,0,0))
image(testImage)
testImage <- binning(testImage,binningFactor = 4)
image(testImage)
## End(Not run)
```

---

 bwApply

*bwApply*


---

**Description**

bwApply allow to get new object from a black / white matrix All NZs at black positions will be taken

**Usage**

```
bwApply(object, bwMatrix)

## S4 method for signature 'MassSpectra,matrix'
bwApply(object, bwMatrix)
```

**Arguments**

object            object of class MassImage  
 bwMatrix        matrix with boolean or numeric 1 and 0

**Value**

object of class MassImage multiplied with B/W matrix

**Examples**

```

library(tofsimsData)
data(tofsimsData)
testImage <- PCAnalysis(testImage, nComp = 2)
library(EBImage)
mask<-thresh(imageMatrix(analysis(testImage,noAccess = 1),comp = 1), w = 15, h = 15)
#inverse of mask
mask <- (mask-1)^2
par(mfcol=c(1,2), oma=c(0,0,0,0), mar=c(0,0,0,0))
image(testImage)
image(bwApply(testImage, mask))

```

---

calibPointNew

*Generic method calibPointNew that modifies slot calibPoints*


---

**Description**

calibPointNew is a method to set a new mass calibration point

**Usage**

```

calibPointNew(object, mz, reset = FALSE, value = NULL)

## S4 method for signature 'MassSpectra,numeric'
calibPointNew(object, mz, reset = FALSE, value = NULL)

```

**Arguments**

object	MassSpectra object
mz	the m/z value to be specified with a TOF value
reset	shall the list of calibration points be reset
value	TOF value to be assigned to mz

**Details**

calibPointNew ia a method to set a new mass calibration point. When value is not provided as arguemnt, the TOF for the chosen mz value has to be chosen interactively by mouse.

**Value**

call by reference, hence MassSpectra object with new calib point  
object MassSpectra with added/updated calibration points

## Examples

```
library(tofsimsData)
data(tofsimsData)
testSpectra <- calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra <- calibPointNew(testSpectra, mz = 181, value = 181.0228)
calibPoints(testSpectra)
par(mfcol=c(1,2))
plot(testSpectra,mzRange=c(38.5,40.5),type='l')
testSpectra <- recalibrate(testSpectra)
plot(testSpectra, mzRange=c(38.5,40.5), type='l')
```

---

calibPoints	calibPoints, <i>slot of MassSpectra class objects</i>
-------------	---

---

## Description

calibPoints, slot of MassSpectra class objects

## Usage

```
calibPoints(object)

calibPoints(object) <- value

## S4 method for signature 'MassSpectra'
calibPoints(object)

## S4 replacement method for signature 'MassSpectra'
calibPoints(object) <- value
```

## Arguments

object	object of class MassSpectra
value	data.frame replacement values for calibPoints slot

## Value

contents of slot calibPoints

## See Also

object [MassSpectra](#) other slots [mz analysis](#) [analysisName](#) [instrument](#) [nz calibration](#)

**Examples**

```

library(tofsimsData)
data(tofsimsData)
testSpectra<-calibPointNew(testSpectra, mz = 15, value = 15.0232)
testSpectra<-calibPointNew(testSpectra, mz = 181, value = 181.0569)
## access 'calibPoint' slot of 'MassSpectra' object
calibPoints(testSpectra)
## replacing values in the 'calibPoint' slot
calibPoints(testSpectra)[2,2]<-297000
calibPoints(testSpectra)

```

---

calibration

calibration, *slot of MassSpectra class objects*


---

**Description**

calibration, slot of MassSpectra class objects

Generic setter for slot calibration<-

**Usage**

```
calibration(object)
```

```
calibration(object) <- value
```

```
## S4 method for signature 'MassSpectra'
```

```
calibration(object)
```

```
## S4 replacement method for signature 'MassSpectra'
```

```
calibration(object) <- value
```

**Arguments**

object            object of class MassSpectra

value            data.frame with replacement values for calibration slot

**Value**

content of calibration slot

**See Also**

object [MassSpectra](#) other slots [mz](#) [analysis](#) [analysisName](#) [instrument](#) [calibPoints](#) [nz](#)

**Examples**

```

library(tofsimsData)
data(tofsimsData)
## access calibration slot
calibration(testSpectra)
## replacing the values of the 'calibration' slot is
## possible but it makes at the moment no sense as it
## doesn't change the actual mass calibration. The
## 'calibration' slot is just used to store the values
## while 'recalibration' uses the values from
## 'calibPoints' slot.
calibration(testSpectra) <- data.frame(intercept = 21420, slope = 20480)
calibration(testSpectra)

```

---

changePeakWidth	<i>method changePeakWidth</i>
-----------------	-------------------------------

---

**Description**

method changePeakWidth

**Usage**

```

changePeakWidth(object, selectMz, lowerWidth, upperWidth, ...)

## S4 method for signature 'PeakList,missing,missing,missing'
changePeakWidth(object, selectMz, lowerWidth, upperWidth, ...)

## S4 method for signature 'PeakList,numeric,numeric,numeric'
changePeakWidth(object, selectMz, lowerWidth, upperWidth, ...)

```

**Arguments**

object	PeakList object
selectMz	numeric change width of peak closest to selectMz
lowerWidth	numeric lower width value in mass units
upperWidth	numeric upper width value in mass units
...	additional args

**Details**

method changePeakWidth is used to modify the peak width of an individual peak it should be called with the argument mzRange to zoom into the region of interest for choosing the peak. Then two further clicks for choosing the (new) lower and upper peak widths.

**Value**

PeakList object with upated peak widths

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testPeakList<-PeakList(analysisName = analysisName(testSpectra),
instrument = instrument(testSpectra),
nz = nz(testSpectra),
calibration = calibration(testSpectra),
calibPoints = calibPoints(testSpectra),
mz = mz(testSpectra),
peakIDs = NULL,
peakMzs = NULL)
par(mfcol=c(1,2))
testPeakList<-addPeaks(testPeakList, mzs=26:31, width=0.4)
peakWidths(testPeakList)
testPeakList<-changePeakWidth(testPeakList, selectMz = 27, lowerWidth = 0.2, upperWidth = 0.3)
peakWidths(testPeakList)
```

---

check.extension	<i>Check file extension</i>
-----------------	-----------------------------

---

**Description**

Function to check file extension

**Usage**

```
check.extension(filepath, extension)
```

**Arguments**

filepath	character
extension	character

**Details**

This function is used for check the file extension

**Value**

boolean

**Author(s)**

Lorenz Gerber, Viet Mai Hoang



---

computeMNF	<i>compute MNF</i>
------------	--------------------

---

**Description**

compute MNF, helper for MNF/nnMNF

**Usage**

```
computeMNF(  
  nzData = NULL,  
  noise = NULL,  
  SNR = NULL,  
  ind = NULL,  
  iter = TRUE,  
  limitSNR = NULL,  
  covNoise = NULL  
)
```

**Arguments**

nzData	matrix
noise	matrix
SNR	numeric
ind	numeric
iter	boolean
limitSNR	numeric
covNoise	matrix

**Details**

This is a helper function for the MNF/nnMNF function and originates from the mzImage package.

**Value**

MNF transform

---

computeNoise	<i>computeNoise</i>
--------------	---------------------

---

**Description**

computeNoise determinates the noise by nearest neighbour estimate. This is a helper function for the nnMNF method.

**Usage**

```
computeNoise(stat, x, y)
```

**Arguments**

stat	unknown
x	unknown
y	unknown

**Details**

computeNoise determinates the noise by nearest neighbour estimate. This is a helper function for the nnMNF method and originates from the mzImage package.

**Value**

matrix numeric noise

---

coordToPixel	<i>coordToPixel</i> <i>coordToPixel</i> translates xy coordinates from the locator() function to cell coordinates from the image function. Origo is according to ToF-SIMS images the upper left corner.
--------------	---

---

**Description**

coordToPixel coordToPixel translates xy coordinates from the locator() function to cell coordinates from the image function. Origo is according to ToF-SIMS images the upper left corner.

**Usage**

```
coordToPixel(object, xy)
```

**Arguments**

object	of class MassImage
xy	numeric vector with x/y locator coordinate

**Value**

xy coordinate of MassImage pixels

coordToPixel,MassImage,numeric-method  
*coordToPixel*

**Description**

coordToPixel

**Usage**

```
## S4 method for signature 'MassImage,numeric'
coordToPixel(object, xy)
```

**Arguments**

object            of class MassImage  
xy                numeric vector with x/y locator coordinate

**Details**

coordToPixel translates xy coordinates from the locator() function to cell coordinates from the image function. Origo is according to ToF-SIMS images the upper left corner.

**Value**

xy coordinate of MassImage pixels

covDiffCalc            *covDiffCalc calculates a x/y shift covariance matrix of a multispectral image according to Switzer and Green 1984.*

**Description**

covDiffCalc calculates a x/y shift covariance matrix of a multispectral image according to Switzer and Green 1984.

**Usage**

```
covDiffCalc(nzData, dataObject)
```

**Arguments**

nzData	unknown
dataObject	unknown

**Value**

shifted cov matrix

---

cReadRawPhi

*Ulvac phi ToF-SIMS raw data import*


---

**Description**

Function to read raw data from the ulvac-phi trift TOF-SIMS

**Usage**

```
cReadRawPhi(
  analysisName,
  mode = c("spectra", "imagepeaks", "image"),
  PeakListobj = c(),
  ...
)
```

**Arguments**

analysisName	character
mode	character
PeakListobj	object of class PeakList
...	additional args

**Details**

This import function works on data recorded on the ulvac-phi trift ToF-SIMS with WinCadence software version V4.2. Other versions mostl likely will not work. In the current version, data has to be imported with 16bit word length, then converted to 64bit binary and finally converted and read with the word lengths of the respective variables. Currently, the data is unit mass binned with bins of size one from -0.5 to + 0.5.

**Value**

parsed rawdata for further processing

**Author(s)**

Lorenz Gerber, Viet Mai Hoang

---

cTable                      *cTable is a C++ implementation to make contingency tables*

---

**Description**

cTable is a C++ implementation to make contingency tables

**Usage**

```
cTable(vect)
```

**Arguments**

vect                      NumericVector

**Value**

vars freqs

---

dim, MassImage-method    *method dim for MassImage*

---

**Description**

method dim for MassImage

**Usage**

```
## S4 method for signature 'MassImage'  
dim(x)
```

**Arguments**

x                          object of class MassImage

**Value**

vector numeric

---

dim,MassSpectra-method

*method definition 'dim' for 'MassSpectra' dim is a primitive*

---

### Description

method definition 'dim' for 'MassSpectra' dim is a primitive

### Usage

```
## S4 method for signature 'MassSpectra'
dim(x)
```

### Arguments

x                    object object of type MassSpectra

### Value

numeric value

---

EigenDecompose

*EigenDecompose for the MNF analysis*

---

### Description

EigenDecompose for the MNF analysis

### Usage

```
EigenDecompose(A, B, startIndex, endIndex)
```

### Arguments

A	NumericMatrix
B	NumericMatric
startIndex	int
endIndex	int

### Value

eigval eigvec mA mB

---

extract.header.data     *extract variable names and values from Ulvac-phi ToF-SIMS datafile headers*

---

**Description**

Extracting the data from a Ulvac-phi Tof-SIMS raw header character string.

**Usage**

```
extract.header.data(header)
```

**Arguments**

header                header as a raw character string

**Details**

This function takes a raw header character string read by `get.raw.header()` as input and extracts variable names and values. values are currently forwarded just as character string. This is a helper function for `read.raw.phi`.

**Value**

list with two vectors containing variable names and values as characters

**Author(s)**

Lorenz Gerber

---

findClosestMatch     *Find single value 'toMatch' in vector 'MatchIn'*

---

**Description**

Find single value 'toMatch' in vector 'MatchIn'

**Usage**

```
findClosestMatch(toMatch, matchIn, twoMatch)
```

**Arguments**

toMatch                numeric  
matchIn                vector numeric  
twoMatch                character 'upper' or 'mean'

**Value**

numeric ID of match

---

findPeakWidth	<i>generic method findPeakWidth</i>
---------------	-------------------------------------

---

**Description**

method findPeakWidth

**Usage**

```
findPeakWidth(
    object,
    p = 3,
    n = 5,
    span = 100,
    widthExtLower = 1.5,
    widthExtUpper = 1.75,
    ...
)

## S4 method for signature 'PeakList'
findPeakWidth(
    object,
    p = 3,
    n = 199,
    span = 100,
    widthExtLower = 1.7,
    widthExtUpper = 2,
    ...
)
```

**Arguments**

object	object of class PeakList
p	numeric value for savitzky-golay filter on first derivate
n	numeric value for savitzky-golay filter on first derivate
span	numeric smoothing for determining local minima/maxima values
widthExtLower	numeric factor to extend lower peak width
widthExtUpper	numeric factor to extend upper peak width
...	additional args



**Details**

This method uses signal processing to determine lower and upper peak width limits based on local max/min detection of the first derivate next to peak center values. The initial code for local min/max detection is adapted from the CRAN package 'alsace'.

**Value**

object of class PeakList with updated peaks

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testPeakList<-PeakList(analysisName = analysisName(testSpectra),
instrument = instrument(testSpectra),
nz = nz(testSpectra),
calibration = calibration(testSpectra),
calibPoints = calibPoints(testSpectra),
mz = mz(testSpectra),
peakIDs = NULL,
peakMzs = NULL)
par(mfcol=c(1,2))
plot(testPeakList, mzRange=c(25,32), type = 'l')
testPeakList<-addPeaks(testPeakList, mzs=26:31, width=0.4)
testPeakList<-findPeakWidth(testPeakList, p = 3, n = 199,
span = 100, widthExtLower = 2, widthExtUpper = 2)
plot(testPeakList, mzRange=c(25,32), type = 'l')
```

---

getTOFs

*generic method to calculate and get TOFs*

---

**Description**

generic method to calculate and get TOFs

method getTOFs

**Usage**

```
getTOFs(object)
```

```
## S4 method for signature 'MassSpectra'
```

```
getTOFs(object)
```

**Arguments**

object            object of class MassSpectra

**Value**

vector with ToFs  
vector numeric with TOF values

**Examples**

```
library(tofsimsData)
data(tofsimsData)
timeOfFlight <- getTOFs(testSpectra)
head(timeOfFlight)
```

---

image	<i>set a generic method for image</i>
-------	---------------------------------------

---

**Description**

Method to visualize an IMS Mass Image of class MassImage

**Usage**

```
image(x, ...)  
  
## S4 method for signature 'MassImage'  
image(x, ..., mzSelect = NULL)  
  
## S4 method for signature 'PCA'  
image(x, comp, ...)
```

**Arguments**

x	object object with image data
...	additional args
mzSelect	vector, which m/z to combine for visualization. if none are chosen, the TIC is shownhel
comp	numeric which component to visualize

**Value**

graphical output  
image plot of the ToF SIMS image data

**Examples**

```
testImage<-MassImage('dummy')
image(testImage)
## Not run:
library(tofsimsData)
data(tofsimsData)
image(testImage)
## End(Not run)
library(tofsimsData)
data(tofsimsData)
testImage<-PCAnalysis(testImage,3)
image(analysis(testImage, 1), comp = 1)
```

---

imageMatrix

*generic method to obtain imageMatrix*

---

**Description**

generic method to obtain imageMatrix

Method imageMatrix for class MassImage

**Usage**

```
imageMatrix(object, ...)
```

```
## S4 method for signature 'MassImage'
imageMatrix(object)
```

```
## S4 method for signature 'PCA'
imageMatrix(object, comp, ...)
```

**Arguments**

object            object of class MassImage

...               additional args

comp              numeric which component

**Value**

numeric matrix

matrix numeric

**Examples**

```

library(tofsimsData)
data(tofsimsData)
## the TIC matrix can be extracted
dataMatrix <- imageMatrix(testImage)
dim(dataMatrix)
## the matrix can be visualized with the
## normal image() function
image(dataMatrix)

```

---

import	<i>import is the C++ code for importing iontof raw data</i>
--------	---

---

**Description**

import is the C++ code for importing iontof raw data

**Usage**

```
import(rFilename, fType, imageSize, upperMass)
```

**Arguments**

rFilename	CharacterVector
fType	CharacterVector
imageSize	int
upperMass	int

**Value**

imported binary raw data

---

import.raw	<i>Raw data import</i>
------------	------------------------

---

**Description**

Function to read raw data.

**Usage**

```

import.raw(
  analysisName,
  mode = c("spectra", "imagepeaks"),
  PeakListobj = c(),
  untilScan = NULL,
  ...
)

```

**Arguments**

analysisName	character
mode	character
PeakListobj	object of class PeakList
untilScan	numeric read data up to which scan number
...	additional args

**Details**

This import function works on GRD and ITZIP format

**Value**

parsed rawdata for further processing

**Author(s)**

Lorenz Gerber, Viet Mai Hoang

---

instrument	instrument, <i>slot of MassSpectra class objects</i>
------------	--

---

**Description**

instrument, slot of MassSpectra class objects

**Usage**

```

instrument(object, ...)

instrument(object) <- value

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

## S4 replacement method for signature 'MassSpectra'
instrument(object) <- value

```

**Arguments**

object	object of class MassSpectra
...	additional args
value	character name of instrument used in the experiment

**Value**

content of instrument slot

**See Also**

object [MassSpectra](#) other slots [mz](#) [analysis](#) [analysisName](#) [nz](#) [calibPoints](#) [calibration](#)

**Examples**

```
library(tofsimsData)
data(tofsimsData)
## access instrument slot in MassSpectra objects
instrument(testSpectra)
## values for the 'instrument' slot can currently be
## 'iontof' or 'ulvacphi'. It is not advisable to
## change those values manually
```

---

iters

*generic accessor for iters slot*

---

**Description**

generic accessor for iters slot

**Usage**

```
iters(object)
```

**Arguments**

object            object of class MCR

**Value**

content of iters slot

---

iters,MCR-method      *MCR accessor iters,*

---

**Description**

MCR accessor iters,

**Usage**

```
## S4 method for signature 'MCR'  
iters(object)
```

**Arguments**

object      object of class MCR

**Value**

iters from object

---

itzipName      *defining generic accessor method for "itzipName"*

---

**Description**

defining generic accessor method for "itzipName"

**Usage**

```
itzipName(object)
```

**Arguments**

object      internal

**Value**

content of itzipName

---

itzipName<-                    *generic for setter itzipName*

---

### Description

generic for setter itzipName

### Usage

```
itzipName(object) <- value
```

### Arguments

object	internal
value	internal

### Value

object with updated itzipName slot

---

LapackGenEigen                *LapackGenEigen*

---

### Description

LapackGenEigen is helper function for MNF and nnMNF

### Usage

```
LapackGenEigen(A, B, IL = 1, IU = 3)
```

### Arguments

A	matrix
B	matrix
IL	int start index
IU	int end index

### Details

LapackGenEigen is adapted from the mzImage package. While it initially used dsygvx from the LAPACK library, it is now ported to RcppArmadillo, using the eig\_pair function.

### Value

list with values, vectors and info



---

legend.col	<i>legend.col</i>
------------	-------------------

---

**Description**

legend.col is a helper for the plot function of Scoreplots. It allows to visualize a third component by a color range. legend.col plots the color range as legend on the side of the plot

**Usage**

```
legend.col(col, lev)
```

**Arguments**

col	character color
lev	character levels

**Value**

graphical output

---

look.for.itzip.property	<i>Get ITZIP property value</i>
-------------------------	---------------------------------

---

**Description**

Function to extract value by passing property name

**Usage**

```
look.for.itzip.property(itzipName, itzipProperties)
```

**Arguments**

itzipName	character
itzipProperties	character

**Details**

This function is used to get ITZIP property value by passing its name

**Value**

character value from itzipProperties corresponding itzipName

**Author(s)**

Lorenz Gerber, Viet Mai Hoang

---

MAF

*Class MAF*

---

**Description**

Class MAF contains methods for Maximum Autocorrelation Factors analysis

MAF is a Maximum Autocorrelation Factor Analysis

**Usage**

```
MAF(dataObject, nComp = 10, usePCA = TRUE)
```

```
MAF(dataObject, nComp = 10, usePCA = TRUE)
```

**Arguments**

dataObject	object of type MassImage
nComp	integer number of components
usePCA	boolean use PCA

**Details**

Class MAF contains methods for Maximum Autocorrelation Factors analysis

MAF is a Maximum Autocorrelation Factor Analysis. The code is implemented from the publication of

**Value**

object of type MAF

**Examples**

```
library(tofsimsData)
data(tofsimsData)
## Not run: data(tofsimsData)
MAF(testImage,5,TRUE)
image(analysis(testImage,1),comp = 1)
## End(Not run)
```

---

makeTIC	<i>generic for makeTIC</i>
---------	----------------------------

---

**Description**

generic for makeTIC

**Usage**

makeTIC(object)

**Arguments**

object            object of type MassSpectra

**Value**

object of class MassSpectra with TIC

---

makeTIC,MassSpectra-method  
*Method makeTIC for MassSpectra Class*

---

**Description**

Method makeTIC sums up all Mass Spectra in the called Mass Spectra object

**Usage**

```
## S4 method for signature 'MassSpectra'  
makeTIC(object)
```

**Arguments**

object            object of class MassSpectra

**Value**

object of class MassSpectra with just one spectra, the TIC

---

manualSelectPeaks	<i>This method is base method for plotting and manual select data</i>
-------------------	---

---

**Description**

This method is base method for plotting and manual select data

**Usage**

```
manualSelectPeaks(object, n = 512, ...)
```

**Arguments**

object	object of type PeakList
n	numeric
...	additional args

**Value**

numeric x coordinates

---

MassImage	<i>Class MassImage</i>
-----------	------------------------

---

**Description**

Class MassImage contains the information to shape a number of mass spectra into an image.

MassImage is also the call to the class constructor. It is used for importing both BIF/BIF6 and raw image data.

**Usage**

```
MassImage(
  select = c("ulvacbif", "iontof bif", "iontofgrdpeaks", "ulvacrawpeaks", "dummy"),
  analysisName,
  PeakListobj = c(),
  untilScan = NULL,
  ...
)
```

```
MassImage(
  select = c("ulvacbif", "iontof bif", "iontofgrdpeaks", "ulvacrawpeaks", "dummy"),
  analysisName,
  PeakListobj = c(),
  untilScan = NULL,
  ...
)
```

**Arguments**

select	character, 'ulvacbif', 'iontof bif', 'iontofgrdpeaks', 'ulvacrawpeaks', 'dummy'
analysisName	character, name of analysis
PeakListobj	PeakList class object, used as peaklist for rawdata import
untilScan	integer or NULL to determine number of ToF-SIMS scans to import
...	additional args

**Details**

Class MassImage inherits from the classes MassAnalysis and MassSpectra. It contains the information to shape a number of mass spectra into an image.

MassImage is the user class constructor to obtain a MassImage object. Data can be imported from BIF or raw data files (Iontof or Ulvacphi). To import raw data, a MassSpectra object with a valid PeakList object has to be provided as argument.

**Value**

object of class MassImage

**Slots**

xy vector giving the pixel dimension of the image

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
# creating dummy data
testImage<-MassImage('dummy')
image(testImage)
## Not run:
# import of rawdata
# first a PeakList object has to be created
library(tofsimsData)
data(tofsimsData)
testSpectra <- calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra <- calibPointNew(testSpectra, mz = 181, value = 181.0228)
testSpectra <- recalibrate(testSpectra)
testSpectra <- unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
# obtaining the path to the raw data file in 'tofsims' package
importFile<-system.file("rawdata", "trift_test_001.RAW", package = "tofsimsData")
rawImportedImage <- MassImage('ulvacrawpeaks', importFile,
PeakListobj = testSpectra)
image(rawImportedImage)

## End(Not run)
```

---

 MassSpectra

 Class MassSpectra
 

---

### Description

Class MassSpectra is the main data container in the tofsims package as it contains the individual mass spectra.

MassSpectra is also the call to class constructor. It is used for importing high-resolution mass spectra from raw data.

### Usage

```
MassSpectra(select = c("ulvacraw", "iontofgrd", "dummy"), analysisName, ...)
```

```
MassSpectra(select = c("ulvacraw", "iontofgrd", "dummy"), analysisName, ...)
```

### Arguments

select	character, 'ulvacraw', 'iontofgrd', 'dummy'
analysisName	character, the (file)name of the dataset
...	additional args

### Details

Class MassSpectra is the main data container of the tofsims package, containing the individual mass spectra in the slot `nz`. Additional metadata about the analysis can be found in the slots `analysisName` and `instrument`. Values for slope and intercept of the linear mass calibration equation are stored in the slot `calibration`. The `M/z` values can be found in `mz`. `calibration` allows calculating from `M/z` values back to times-of-flight. The slot `calibPoints` is used to recalibrate the dataset. It contains a `data.frame` with the columns `mz` and `TOF`. The slot `analysis` of type `list`, is used as a container for data analysis objects. Typically, object of the class MassSpectra are constructed during data import using the user constructor function with the same name as the class, `MassSpectra`.

MassSpectra is also the call to class constructor. It is used for importing high-resolution mass spectra from raw data.

### Value

object of class MassSpectra

### Slots

analysisName	character vector with the import filename
instrument	character vector type of instrument used in the experiment
calibration	data frame for numerics slope and intercept of the mass calibration

calibPoints data frame for time of flight to maass to charge calibration  
 nz matrix with rows of ion counts and columns as toftimes or mass to charge ratios  
 mz vector same length as columns in nz for mass to charge values

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
## Not run:
## access rawdata in tofsims package
library(tofsimsData)
importFile<-system.file("rawdata", "trift_test_001.RAW", package = "tofsimsData")
MassSpectra('ulvacraw', importFile)

## End(Not run)
## create dummy MassSpectra object
MassSpectra('dummy')
```

---

MCR-class

*Class MCR*


---

**Description**

Class MCR contains methods for 'Multivariate Curve Resolution by Alternate Least Squares'  
 opaMCR is a MCR-ALS function using the Orthogonal Projection Approach from

**Usage**

```
opaMCR(dataObject, opaComps, maxiter = 10)
```

**Arguments**

dataObject	object of class MassImage
opaComps	numeric number of components for the opa method
maxiter	numeric how many iterations

**Details**

Class MCR contains methods for 'Multivariate Curve Resolution by Alternate Least Squares'  
 opaMCR uses the function `alsace::opa()` (Orthogonal Projection Approach, Bioconductor package 'alsace') for start estimates of pure spectras and `ALS::als()` (CRAN package 'ALS') as MCR-ALS implementation. This method is doing fine with images up to 256x256 pixels. For larger images, memory usage becomes unreasonably high.

**Value**

object of class MCR

**Slots**

RSS numeric residual sum of squares

resids matrix with residuals

iters numeric number of iterations

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
testImage<-MassImage('dummy')
testImage<-opaMCR(testImage, 2, 2)
image(analysis(testImage,1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
testImage<-MCR(testImage, 5, 5)
image(analysis(testImage,1), comp = 1)

## End(Not run)
```

---

MNF

*Class MNF*

---

**Description**

Class MNF contains methods for Maximum Autocorrelation Factors analysis

This method calculates MNF transform using the diagonal shift method from Switzer and Green (1984) to estimate the noise.

**Usage**

```
MNF(dataObject)
```

```
MNF(dataObject)
```

**Arguments**

dataObject      object of type massImage



## Details

Class MNF contains methods for Maximum Autocorrelation Factors analysis

Minimum Noise Fraction according Green et al. (1988) using diagonal shift method from Switzer and Green (1984) to estimate the noise. As the original package mzImage from Stone et al. 2012 is no longer maintained, we use it as code base for the present version. The C code was implemented through Rcpp (Eddelbuettel and Francois, 2011). Practically, this method uses `covDiffCalc` from the MAF method. The present function is a user constructor that will create a new analysis slot in the chosen `MassSpectra/MassImage` object.

## Value

object of class MNF

## Examples

```
testImage<-MassImage('dummy')
testImage<-MNF(testImage)
image(analysis(testImage,1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
MNF(testImage)
image(analysis(testImage,1), comp = 1)
## End(Not run)
```

---

mz,MassSpectra-method *mz getter method*

---

## Description

mz getter method

mz setter method

## Usage

```
## S4 method for signature 'MassSpectra'
mz(object)
```

```
## S4 replacement method for signature 'MassSpectra'
mz(object) <- value
```

## Arguments

object            of type `MassSpectra`

value            double mass to charge ratio

**Value**

MassSpectra object with updated mz slot

**Examples**

```
library(tofsimsData)
data(tofsimsData)
## access the mz values fo each spectra point
mz(testSpectra)[1:100]
## replace a mz value
mz(testSpectra)[1] <- 0.000025
mz(testSpectra)[1:100]
```

---

nComp	<i>generic accessor method for slot nComp</i>
-------	---

---

**Description**

generic accessor method for slot nComp  
PCA accessor nComp, number of component

**Usage**

```
nComp(object)

## S4 method for signature 'PCA'
nComp(object)
```

**Arguments**

object            object of class PCA

**Value**

contents of nComp slot  
nuemric number of components

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testImage<-PCAnalysis(testImage,4)
nComp(analysis(testImage,1))
```

---

ndim                      *generic accessor method for slot ndim*

---

**Description**

generic accessor method for slot ndim

**Usage**

ndim(object)

**Arguments**

object                      object of class MassSpectra

**Value**

contents of slot ndim

---

ndim,MassSpectra-method  
*method definition 'ndim' on 'MassSpectra'*

---

**Description**

method definition 'ndim' on 'MassSpectra'

**Usage**

```
## S4 method for signature 'MassSpectra'  
ndim(object)
```

**Arguments**

object                      object of type MassSpectra

**Value**

numeric value

---

nearestNeighbourMean    *nearestNeighbourMean*

---

**Description**

nearestNeighbourMean helper for nnMNF

**Usage**

nearestNeighbourMean(x)

**Arguments**

x                      unknown see mzimage

**Details**

function from mzimage to calculate nearest neighbour means

**Value**

matrix numeric nearest neighbours

---

nnMean                      *nnMean is C++ code for calculating nearest neighbour means in a 2D matrix*

---

**Description**

nnMean is C++ code for calculating nearest neighbour means in a 2D matrix

**Usage**

nnMean(y, nrows, ncols)

**Arguments**

y                      NumericVector  
nrows                  int  
ncols                  int

**Value**

eY

---

`nnMNF`*Class nnMNF*

---

**Description**

Class nnMNF contains methods for Maximum Autocorrelation Factors analysis

This method calculates MNF transform using an nearest neighbour estimate as implemented in `mzImage` from Stone et al. (2012).

**Usage**

```
nnMNF(dataObject, limitsNR = 1.5)
```

```
nnMNF(dataObject, limitsNR = 1.5)
```

**Arguments**

`dataObject`      object of type `MassImage`

`limitsNR`        numeric

**Details**

Class nnMNF contains methods for Maximum Autocorrelation Factors analysis

Minimum Noise Fraction according Green et al. (1988) but using a nearest neighbour estimate for the noise determination as seen in the package `mzImage` from Stone et al. (2012). As the mentioned package is no longer maintained, we used an archived version as code base for a new version. The C code was implemented through Rcpp (Eddelbuettel and Francois, 2011). The present function is a user constructor that will create a new analysis slot in the chosen `MassSpectra/MassImage` object.

**Value**

object of class MNF

**Examples**

```
testImage<-MassImage('dummy')
testImage<-MNF(testImage)
image(analysis(testImage,1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
testImage<-nnMNF(testImage)
image(analysis(testImage,1), comp = 1)
## End(Not run)
```

---

noPlottingData	<i>generic method for 'noPlottingData' aka 'is.null'</i>
----------------	--

---

**Description**

generic method for 'noPlottingData' aka 'is.null'

**Usage**

```
noPlottingData(object)
```

**Arguments**

object            object of class PCA

**Value**

boolean validity check of PCA object

---

noPlottingData,PCA-method	<i>Check NULL PCA object</i>
---------------------------	------------------------------

---

**Description**

Check NULL PCA object

**Usage**

```
## S4 method for signature 'PCA'
noPlottingData(object)
```

**Arguments**

object            object of class PCA

**Value**

boolean validity check of class PCA object

---

nPeaks	<i>generic method for nPeaks</i>
--------	----------------------------------

---

**Description**

generic method for nPeaks  
nPeaks accessor/getter nPeaks for PeakList Class

**Usage**

```
nPeaks(object)  
  
## S4 method for signature 'PeakList'  
nPeaks(object)
```

**Arguments**

object            object of class PeakList

**Value**

integer value for number of peaks

**Examples**

```
library(tofsimsData)  
data(tofsimsData)  
testSpectra<-calibPointNew(testSpectra, mz = 15, value = 15.01551)  
testSpectra<-calibPointNew(testSpectra, mz = 181, value = 181.0228)  
testSpectra<-recalibrate(testSpectra)  
testSpectra<-unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),  
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))  
nPeaks(testSpectra)
```

---

nz	<i>nz, slot of MassSpectra class objects</i>
----	--

---

**Description**

nz, slot of MassSpectra class objects

**Usage**

```
nz(object, mzRange = NULL)

nz(object) <- value

## S4 method for signature 'MassSpectra,missing'
nz(object, mzRange = NULL)

## S4 method for signature 'MassSpectra,numeric'
nz(object, mzRange = NULL)

## S4 replacement method for signature 'MassSpectra'
nz(object) <- value
```

**Arguments**

object	object of class MassSpectra
mzRange	vector numeric mass values for nz matrix
value	matrix replacement values for nz

**Value**

numeric matrix, content of nz

**See Also**

object [MassSpectra](#) other slots [mz analysis](#) [analysisName](#) [instrument](#) [calibPoints](#) [calibration](#)

**Examples**

```
library(tofsimsData)
data(tofsimsData)
## access main data slot
nz(testSpectra)[,1:1000]
```

---

overlayPlot

*generic overlayPlot*

---

**Description**

This function takes as input a list with objects of type MassSpectra. The easiest way to obtain the input data, is to use `mclapply` from the parallel package.



**Usage**

```
overlayPlot(objectList, ...)

## S4 method for signature 'list'
overlayPlot(
  objectList,
  ...,
  type = "l",
  mzRange = c(1, 200),
  PeakListObj = NULL,
  cex.legend = 0.5
)
```

**Arguments**

objectList	list with object of type MassSpectra
...	additional args
type	character type of plot, usually 'l'
mzRange	vector numeric lower and upper range for plotting the spectra
PeakListObj	object a PeakList object can be provided to plot peaks
cex.legend	numeric text size

**Value**

graphical output  
graphical output

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
library(tofsimsData)
data('tofsimsData')
overlayPlot(list(testImage, testSpectra))
```

---

parIndicesSearch      *helper function for parallel processing in rawdata import routines*

---

**Description**

helper function for parallel processing in rawdata import routines

**Usage**

```
parIndicesSearch(rawVector, mzs, mzsOrder, startOrEnd = "start")
```

**Arguments**

rawVector	unknown
mzs	unknown
mzsOrder	unknown
startOrEnd	character 'start' or 'end'

**Value**

numeric indices of time of flight

---

 PCA-class

*Class PCA*


---

**Description**

Class PCA is a virtual class for PCA that will be inherited.

**Details**

Class PCA is a virtual class for PCA that will be inherited.

**Slots**

pcaLoadings matrix that holds the loadings of a principal component like analysis

pcaScores matrix that holds the scores of a principal component like analysis

nComp numeric number of components in the principal component like analysis

imageDim vector x and y values of the image dimension

classOfData character a more detailed description of the analysis type

---

pcaLoadings	<i>generic accessor for slot pcaLoadings</i>
-------------	--

---

**Description**

generic accessor for slot pcaLoadings  
PCA accessor pcaLoadings, loading matrix  
PCA accessor pcaLoadings, loading matrix

**Usage**

```
pcaLoadings(object, comps = c(1, 2))  
  
## S4 method for signature 'PCA,missing'  
pcaLoadings(object)  
  
## S4 method for signature 'PCA,numeric'  
pcaLoadings(object, comps = c(1, 2))
```

**Arguments**

object	object of class PCA
comps	numeric number of components

**Value**

contents of slot pcaLoadings  
matrix numeric with loadings  
vector or matrix numeric with loadings according comps

**Examples**

```
library(tofsimsData)  
data(tofsimsData)  
testImage<-PCAnalysis(testImage,4)  
plot(pcaLoadings(analysis(testImage,1), comps = c(1,2)))
```

pcaMAF *helper function for MAF calculation*

---

**Description**

helper function for MAF calculation

**Usage**

```
pcaMAF(X, nComp)
```

**Arguments**

X	matrix numeric, matrix to calculate PCA from
nComp	number of components

**Value**

principal component analysis

---

PCAnalysis *Class PCAnalysis*

---

**Description**

Class PCAnalysis contains methods for simple PCA analysis  
PCAnalysis is a PCA constructor function

**Usage**

```
PCAnalysis(dataObject, nComp, ...)
```

```
PCAnalysis(dataObject, nComp, ...)
```

**Arguments**

dataObject	object of type MassImage
nComp	integer number of components
...	further args

**Details**

Class PCAnalysis contains methods for simple PCA analysis  
PCAnalysis constructor function uses call by reference. The new object is put into the analysis slot of the dataObject on which PCA was calculated.

**Value**

PCAnalysis class object

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
testImage<-MassImage('dummy')
testImage<-PCAnalysis(testImage, 4)
image(analysis(testImage, 1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
testImage<-PCAnalysis(testImage, nComp = 4)
image(analysis(testImage, 1), comp = 1)
## End(Not run)
```

---

pcaScores

*generic accessor for slot pcaScores*

---

**Description**

generic accessor for slot pcaScores  
PCA accessor pcaScores, pcaScores matrix  
PCA accessor pcaScores, pcaScores matrix

**Usage**

```
pcaScores(object, comps = c(1, 2))

## S4 method for signature 'PCA,ANY'
pcaScores(object)

## S4 method for signature 'PCA,numeric'
pcaScores(object, comps = c(1, 2))
```

**Arguments**

object            object of class PCA  
comps            numeric number of components

**Value**

contents of slot pcaScores  
vector or matrix numeric with scores according comps

## Examples

```
library(tofsimsData)
data(tofsimsData)
testImage<-PCAnalysis(testImage,4)
plot(pcaScores(analysis(testImage,1), comps = c(1,2)))
```

---

peakIDs	peakIDs, <i>slot of PeakList class objects</i>
---------	--

---

## Description

peakIDs, slot of PeakList class objects

## Usage

```
peakIDs(object)

peakIDs(object) <- value

## S4 method for signature 'PeakList'
peakIDs(object)

## S4 replacement method for signature 'PeakList'
peakIDs(object) <- value
```

## Arguments

object	object of class PeakList
value	data.frame

## Value

content of slot peakIDs

## Examples

```
library(tofsimsData)
data(tofsimsData)
testSpectra<-calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra<-calibPointNew(testSpectra, mz = 181, value = 181.0228)
testSpectra<-recalibrate(testSpectra)
testSpectra<-unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
peakIDs(testSpectra)[,1:10]
```

---

PeakList	<i>Class PeakList</i>
----------	-----------------------

---

### Description

Class PeakList is an extension of TIC class that can hold information about peaks.

Class PeakList inherits from the classes MassAnalysis, MassSpectra and TIC.

PeakList class constructor

### Usage

```
PeakList(
  analysisName = NULL,
  instrument = NULL,
  nz = NULL,
  calibration = NULL,
  calibPoints = NULL,
  mz = NULL,
  peakIDs = NULL,
  peakMzs = NULL,
  ...
)
```

```
PeakList(
  analysisName = NULL,
  instrument = NULL,
  nz = NULL,
  calibration = NULL,
  calibPoints = NULL,
  mz = NULL,
  peakIDs = NULL,
  peakMzs = NULL,
  ...
)
```

### Arguments

analysisName	character vector with the import filename
instrument	character vector type of instrument used in the experiment
nz	matrix numeric containing ion counts, rows are image points, column toftimes/mass to charge ratios
calibration	data frame for numerics slope and intercept of the mass calibration
calibPoints	data frame for time of flight to maass to charge calibration
mz	vector same length as columns in nz for mass to charge values
peakIDs	matrix integer ID for peaks

peakMzs        matrix with mass to charge values for lower, middle and upper peak values  
 ...            additional args

### Details

The PeakList class constructor is used to construct a new PeakList object. Input are currently all needed variables.

### Value

object of class PeakList

### Slots

peakIDs matrix integer ID for peaks  
 peakMzs matrix with mass to charge values for lower, middle and upper peak values

### Author(s)

Lorenz Gerber <lorenz.gerber@slu.se>

Lorenz Gerber <lorenz.gerber@slu.se>

### Examples

```
# The typical way to obtain a PeakList object is by
# applying some peak picking method to a MassSpectra
# below an example using the 'unitMassPeaks' method
library(tofsimsData)
data(tofsimsData)
testSpectra<-calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra<-calibPointNew(testSpectra, mz = 181, value = 181.0228)
testSpectra<-recalibrate(testSpectra)
testSpectra<-unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
show(testSpectra)
```

---

peakMzs                            peakMzs, *slot of PeakList class objects*

---

### Description

peakMzs, slot of PeakList class objects



**Usage**

```
peakMzs(object)

peakMzs(object) <- value

## S4 method for signature 'PeakList'
peakMzs(object)

## S4 replacement method for signature 'PeakList'
peakMzs(object) <- value
```

**Arguments**

object	object of class PeakList
value	data.frame

**Value**

contents of slot peakMzs

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectra<-calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra<-calibPointNew(testSpectra, mz = 181, value = 181.0228)
testSpectra<-recalibrate(testSpectra)
testSpectra<-unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
peakMzs(testSpectra)[,1:10]
```

---

peakPick

*generic method peak.pick*

---

**Description**

method peakPick

**Usage**

```
peakPick(object, span = 100, ...)

## S4 method for signature 'MassSpectra'
peakPick(object, span = 100, ...)
```

**Arguments**

object	object of class MassSpectra
span	numeric parameter for local max/min detection
...	additional args

**Details**

Method peakPick for MassSpectra class, works as a constructor for PeakList class. The local min/max detection implementation is adapted from the CRAN package 'alsace'.

**Value**

object of class PeakList with updated slots PeakIDs and peakMzs  
 object of class PeakList

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectra <- reduceSpectrumResolution(object = testSpectra, everyN = 4, mode = 'keep')
testSpectra <- smootherSpline(testSpectra, stepsize = 10, spar = 0.3)
testSpectra <- smootherGolay(testSpectra, p = 3, n = 5)
testSpectra <- peakPick(testSpectra, span = 100)
plot(testSpectra, , mzRange=c(38.5,40.5), type = 'l')
```

---

peaks2Spectra	<i>generic method peaks2Spectra</i>
---------------	-------------------------------------

---

**Description**

peaks2Spectra allows to transfer the peaks from a PeakList object onto a MassSpectra object. By this, the MassSpectra object is promoted into a PeakList object

**Usage**

```
peaks2Spectra(objectPeaks, objectSpectra)

## S4 method for signature 'PeakList,MassSpectra'
peaks2Spectra(objectPeaks, objectSpectra)
```

**Arguments**

objectPeaks	object object of class PeakList
objectSpectra	object object of class MassSpectra

**Value**

object of class PeakList

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectra<-reduceSpectrumResolution(testSpectra, everyN = 4, mode = 'keep')
peakPickSpectra<-testSpectra
peakPickSpectra<-calibPointNew(peakPickSpectra, mz = 15, value = 15.01551)
peakPickSpectra<-calibPointNew(peakPickSpectra, mz = 181, value = 181.0228)
peakPickSpectra<-recalibrate(peakPickSpectra)
peakPickSpectra<-unitMassPeaks(peakPickSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
par(mfcol = c(1,2))
plot(testSpectra, mzRange = c(38.5, 40.5), type = 'l')
testSpectra<-peaks2Spectra(peakPickSpectra, testSpectra)
plot(testSpectra, mzRange = c(38.5, 40.5), type = 'l')
```

---

peakWidths

*Generic method peakWidths*

---

**Description**

Generic method peakWidths

peakWidths

**Usage**

```
peakWidths(object, plot = FALSE)
```

```
## S4 method for signature 'PeakList'
```

```
peakWidths(object, plot = FALSE)
```

**Arguments**

object	PeakList object
plot	boolean should there be graphical output

**Details**

This method will calculate peak widths (m/z) based on lower and upper widths.

Method to return the peakWidth values of all peaks. On plot=TRUE the width values are plotted against the M/z of the corresponding peak.

**Value**

vector of peak widths

**Examples**

```

library(tofsimsData)
data(tofsimsData)
testPeakList<-PeakList(analysisName = analysisName(testSpectra),
instrument = instrument(testSpectra),
nz = nz(testSpectra),
calibration = calibration(testSpectra),
calibPoints = calibPoints(testSpectra),
mz = mz(testSpectra),
peakIDs = NULL,
peakMzs = NULL)
testPeakList<-addPeaks(testPeakList, mzs=26:31, width=0.4)
testPeakList<-findPeakWidth(testPeakList, p = 3, n = 199,
span = 100, widthExtLower = 2, widthExtUpper = 2)
testPeakList<-peakWidths(testPeakList, plot = FALSE)

```

---

plot

*Generic method for plot*


---

**Description**

Method defining plot() for the MassSpectra class plot has no generic by default

**Usage**

```

plot(x, y, ...)

## S4 method for signature 'MassSpectra,missing'
plot(x, y, ..., mzRange = c(0, 200), normalize = FALSE)

## S4 method for signature 'PCA,ANY'
plot(
  x,
  ...,
  comps = c(1, 2),
  pcType = "pcaLoadings",
  label = FALSE,
  labelThreshold = 1
)

```

**Arguments**

x	object of type MassSpectra
y	missing
...	further args
mzRange	vector or length two, indicating the mz range to be plotted
normalize	boolean should the mass spectra be normalized

comps	numeric vector of length two denominating the components to be plotted
pcType	character 'pcaLoadings' or 'pcaScores'
label	boolean plot label
labelThreshold	numeric threshold on which values to plot a label

### Details

The output of this method is adapted for plotting mass spectra. Uncalibrated data is plotted as xy plot while calibrated data is plotted as barplot. The parameter `mzRange` allows choosing the plot range directly according to the `mz` number (when calibrated). The argument `lineplot`, TRUE by default, allows to switch between line and barplot.

### Value

graphical output  
 plot of mass spectra  
 scatter loading/score plot

### Examples

```
## plot method for MassSpectra objects
library(tofsimsData)
data(tofsimsData)
plot(testSpectra, mzRange=c(1,300),type='l')
```

---

plot,MassImage,missing-method  
*Method plot() for MassImage*

---

### Description

Method defining `plot()` for the `MassImage` class plot has no generic by default

### Usage

```
## S4 method for signature 'MassImage,missing'
plot(x, y, ..., mzRange = c(0, 200), normalize = FALSE)
```

### Arguments

<code>x</code>	object of type <code>MassImage</code>
<code>y</code>	missing
<code>...</code>	additional args
<code>mzRange</code>	vector or length two, indicating the <code>mz</code> range to be plotted
<code>normalize</code>	should the mass spectra be normalized

**Details**

This method will call plot method of MassSpectra class.

**Value**

scatter plot with loading or scores

---

plot,PeakList,missing-method  
*Method plot() for MassSpectra*

---

**Description**

Method defining plot() for the MassSpectra class plot has no generic by default

**Usage**

```
## S4 method for signature 'PeakList,missing'
plot(
  x,
  y,
  ...,
  mzRange = c(0, 200),
  plotDeriv = FALSE,
  plotPeaks = TRUE,
  plotWidths = TRUE
)
```

**Arguments**

x	object of type PeakList
y	missing
...	further args
mzRange	vector or length two, indicating the mz range to be plotted
plotDeriv	boolean plot derivate if available
plotPeaks	boolean plot peaks if available
plotWidths	boolean plot peak widths if available

**Details**

The output of this method is adapted for plotting mass spectra. Uncalibrated data is plotted as xy plot while uncalibrated data is plotted as barplot. The parameter mzRange allows choosing the plot range directly according to the mz number (when calibrated).

**Value**

plot spectra with peaks and peak widths

---

points	<i>generic method points generic method points</i>
--------	--

---

### Description

Method defining points() for the MassSpectra class points has no generic by default

### Usage

```
points(x, ...)  
  
## S4 method for signature 'MassSpectra'  
points(x, y, ..., mzRange = c(0, 200), normalize = FALSE)
```

### Arguments

x	vector with mz for mass spectra plot
...	additional args
y	vector with ion counts for mass spectra plot
mzRange	vector of length 2, indicating the mz range to be plotted
normalize	boolean should the mass spectra be normalized

### Details

This function can be used to visualize several spectra in the same plot.

### Value

graphical output  
graphic output

### Examples

```
library(tofsimsData)  
data("tofsimsData")  
plot(testImage, type='l', normalize = TRUE, col = 'blue')  
points(testSpectra, type = 'l', normalize = TRUE, col = 'red')
```

poissonScaling            *generic method for "poissonScaling"*

---

### Description

Poisson scaling for data matrices.

### Usage

```
poissonScaling(object, offset = 1, ...)
```

```
## S4 method for signature 'MassSpectra'  
poissonScaling(object, offset = 1, ...)
```

### Arguments

object	object of class MassSpectra
offset	numeric value for poisson scaling
...	further args

### Details

Poisson scaling is proposed as the method of choice for ToF-SIMS data see Keenan and Kotula 2004. This implementation was done according to a description in Multivariate Analysis of SIMS spectra in ToF-SIMS: Materials Analysis by Mass Spectrometry, Vickerman and Briggs 2013 and the eigenvector wiki. The offset is described in the eigenvector wiki.

### Value

object of class MassSpectra with poisson scaled mass spectra in slot nz  
object of class MassSpectra

### Author(s)

Lorenz Gerber <lorenz.gerber@slu.se>

### Examples

```
## poisson scaling of MassSpectra objects  
testImage <- MassImage('dummy')  
testImage <- poissonScaling(testImage)  
## Not run:  
# poisson scaling on real data  
library(tofsimsData)  
data(tofsimsData)  
par(mfcol=c(2,2))  
plot(testImage, type='l')  
image(testImage)
```



```
testImage <- poissonScaling(testImage)
plot(testImage,type='l')
image(testImage)

## End(Not run)
```

---

PrComp-class

*Class PrComp*

---

### Description

Class PrComp is a wrapper for the S3 function prcomp

PrComp is a PCA constructor function

### Usage

```
prComp(dataObject, ...)
```

### Arguments

dataObject      object of class MassSpectra

...              additional args for prcomp

### Details

Class PrComp is a wrapper for the S3 function prcomp

PrComp constructor function uses call by reference. The new object is put into the analysis slot of the dataObject on which PCA was calculated.

### Value

object of class PrComp

### Slots

scale logical see description of stats::prcomp

center vector see description of stats::prcomp

sdev vector see description of stats::prcomp

### Author(s)

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```

testImage<-MassImage('dummy')
testImage<-prComp(testImage)
image(analysis(testImage, 1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
testImage<-prComp(testImage)
image(analysis(testImage, 1), comp = 1)
## End(Not run)

```

---

PrinComp-class

*Class PrinComp*


---

**Description**

Class PrinComp is a wrapper for the S3 function princomp  
PrinComp is a PCA constructor function

**Usage**

```
prinComp(dataObject, ...)
```

**Arguments**

dataObject	object of class MassSpectra
...	additional args

**Details**

Class PrinComp is a wrapper for the S3 function princomp  
PrinComp constructor function uses call by reference. The new object is put into the analysis slot of the dataObject on which PCA was calculated.

**Value**

object of class prinComp

**Slots**

scale vector see description of stats::princomp  
n.obs numeric see description of stats::princomp  
call language see description of stats::princomp  
center center see description of stats::princomp  
sdev vector see description of stats::princomp

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
testImage <- MassImage('dummy')
testImage<-prinComp(testImage)
image(analysis(testImage, 1), comp = 1)
## Not run:
library(tofsimsData)
data(tofsimsData)
testImage<-prinComp(testImage)
image(analysis(testImage), 1), comp = 1)
## End(Not run)
```

---

readBIF

*ToF-SIMS BIF/BIF6 file import*

---

**Description**

Function to read ToF-SIMS data in the form of preprocessed BIF files

**Usage**

```
readBIF(
  analysisName,
  instrument = c("iontof", "ulvacphi"),
  mode = c("spectra", "image")
)
```

**Arguments**

analysisName : filename of BIF/BIF6 file to read  
instrument : character, 'iontof' or 'ulvacphi'  
mode, 'spectra' or 'image'

**Details**

This function imports BIF files from IONTOF Surface Lab or ULVAC-PHI's WinCadence. This function reads the data sequential directly from the binary stream. Therefore it's rather slow, but uses less memory than the readBIFParallel function.

**Value**

object of type MassImage or MassSpectra

**Author(s)**

Lorenz Gerber

---

recalibrate	<i>Generic method recalibrate</i>
-------------	-----------------------------------

---

**Description**

Generic method recalibrate  
 method recalibrate

**Usage**

```
recalibrate(object)

## S4 method for signature 'MassSpectra'
recalibrate(object)
```

**Arguments**

object            object of class MassSpectra

**Value**

object of class MassSpectra, recalibrated using the data from slots calibPoints  
 object of class MassSpectra, recalibrated mass values

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectra <- calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra <- calibPointNew(testSpectra, mz = 181, value = 181.0228)
calibPoints(testSpectra)
par(mfcol=c(1,2))
plot(testSpectra,mzRange=c(38.5,40.5),type='l')
testSpectra <- recalibrate(testSpectra)
plot(testSpectra, mzRange=c(38.5,40.5), type='l')
```

---

reduceSpectrumResolution	<i>generic method reduceSpectrumResolution</i>
--------------------------	--

---

**Description**

reduceSpectrumResolution

**Usage**

```

reduceSpectrumResolution(object, everyN = 2, mode = "remove")

## S4 method for signature 'MassSpectra'
reduceSpectrumResolution(object, everyN = 2, mode = "remove")

```

**Arguments**

object	object of class MassSpectra
everyN	numeric act on every nth spectra point
mode	character 'remove' or 'keep'

**Details**

The method reduceSpectrumResolution for MassSpectra is used sometimes for performance reasons.

**Value**

object of class MassSpectra with reduced spectral resolution  
object of class MassSpectra

**Examples**

```

library(tofsimsData)
data(tofsimsData)
par(mfcol=c(1,2))
plot(testSpectra, mzRange = c(40,50), type='l')
testSpectra <- reduceSpectrumResolution(object = testSpectra, everyN = 2, mode = 'remove')
plot(testSpectra, mzRange = c(40,50), type='l')

```

---

removePeaks	<i>generic method removePeaks</i>
-------------	-----------------------------------

---

**Description**

generic method removePeaks

removePeaks for PeakList Class allows removing peaks below a certain treshhold of ioncounts. the threshold is not calculated as area, but just from the peak height (ion count at peak center)

removePeaks for PeakList Class allows removing peaks manually

removePeaks for PeakList Class allows removing peaks manually

**Usage**

```

removePeaks(object, mzs, operator, limit, nLocator, ...)

## S4 method for signature 'PeakList,missing,missing,numeric,missing'
removePeaks(object, mzs, operator, limit, nLocator, ...)

## S4 method for signature 'PeakList,missing,missing,missing,numeric'
removePeaks(object, mzs, operator, limit, nLocator, ...)

## S4 method for signature 'PeakList,numeric,missing,missing,missing'
removePeaks(object, mzs, operator, limit, nLocator, ...)

## S4 method for signature 'PeakList,missing,character,numeric,missing'
removePeaks(object, mzs, operator, limit, nLocator, ...)

```

**Arguments**

object	object of class PeakList
mzs	M/z's of peaks to be removed
operator	Accept ">", "<", "==", "<=", ">=", "!="
limit	numeric limit for peaks to be removed
nLocator	numeric how many peaks to remove with visual selection
...	additional args

**Value**

object of class PeakList with removed/updated peaks

**Examples**

```

library(tofsimsData)
data(tofsimsData)
testPeakList<-PeakList(analysisName = analysisName(testSpectra),
instrument = instrument(testSpectra),
nz = nz(testSpectra),
calibration = calibration(testSpectra),
calibPoints = calibPoints(testSpectra),
mz = mz(testSpectra),
peakIDs = NULL,
peakMzs = NULL)
par(mfcol=c(1,2))
testPeakList<-addPeaks(testPeakList, mzs = 26:31, width=0.4)
plot(testPeakList, mzRange = c(25,32), type = 'l')
testPeakList<-removePeaks(testPeakList, mzs = 27)
plot(testPeakList, mzRange = c(25,32), type = 'l')

```

---

resids	<i>generic accessor method for resids</i>
--------	---

---

**Description**

generic accessor method for resids

**Usage**

```
resids(object)
```

**Arguments**

object            object of class MCR

**Value**

content of slot resids

---

resids, MCR-method	<i>MCR accessor resids,</i>
--------------------	-----------------------------

---

**Description**

MCR accessor resids,

**Usage**

```
## S4 method for signature 'MCR'  
resids(object)
```

**Arguments**

object            object of class MCR

**Value**

resids from object

---

RSS	<i>generic accessor for RSS</i>
-----	---------------------------------

---

**Description**

generic accessor for RSS

**Usage**

RSS(object)

**Arguments**

object            object of class MCR

**Value**

content of slot RSS

---

RSS, MCR-method	<i>MCR accessor RSS,</i>
-----------------	--------------------------

---

**Description**

MCR accessor RSS,

**Usage**

```
## S4 method for signature 'MCR'
RSS(object)
```

**Arguments**

object            object of type MCR

**Value**

RSS from object



---

scale	<i>generic for scale</i>
-------	--------------------------

---

### Description

scale autoscaling method for MassSpectra object. Scaling is along the mass channels. Therefore more than one spectra is needed for scaling.

### Usage

```
scale(x, center = TRUE, scale = TRUE)

## S4 method for signature 'MassSpectra'
scale(x, center = TRUE, scale = TRUE)
```

### Arguments

x	object object of class MassSpectra
center	boolean should data be centered
scale	boolean should data be scaled

### Value

object of class MassSpectra with scaled mass spectra  
object of class MassSpectra

### Examples

```
## autoscaling of dummy image data
testImage<-MassImage('dummy')
par(mfcol=c(2,2))
plot(testImage,type='l')
image(testImage)
testImage <- scale(testImage)
plot(testImage,type='l')
image(testImage)
## Not run:
## autoscaling of real spectral data
library(tofsimsData)
data(tofsimsData)
par(mfcol=c(2,2))
plot(testImage,type='l')
image(testImage)
testImage <- scale(testImage)
plot(testImage,type='l')
image(testImage)
## End(Not run)
```

---

show,MassImage-method *method definition 'show' on 'MassImage' show has a generic by default*

---

**Description**

method definition 'show' on 'MassImage' show has a generic by default

**Usage**

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

**Arguments**

object                    object of class MassImage

**Value**

data.frame character

---

show,MassSpectra-method  
*method defining show() for the MassSpectra class show has a generic by default*

---

**Description**

method defining show() for the MassSpectra class show has a generic by default

**Usage**

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

**Arguments**

object                    object of class MassSpectra

**Value**

data.frame character

---

show,PeakList-method    *method defining show() for the MassSpectra class show has a generic by default*

---

**Description**

method defining show() for the MassSpectra class show has a generic by default

**Usage**

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

**Arguments**

object                    object of class PeakList

**Value**

data.frame character

---

smootherGolay            *generic method smootherGolay*

---

**Description**

generic method smootherGolay  
Method smootherGolay for MassSpectra class

**Usage**

```
smootherGolay(object, p = 3, n = 5, ...)

## S4 method for signature 'MassSpectra'
smootherGolay(object, p = 3, n = 5, ...)
```

**Arguments**

object                    object of class MassSpectra  
p                          numeric parameter for savitzky-golay filter  
n                          numeric parameter for savitzky-golay filter  
...                        additional args

**Value**

object of class MassSpectra with updated mass spectra  
object of class MassSpectra with smoothed TIC

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectraSmooth <- smootherGolay(testSpectra, p = 3, n = 9)
overlayPlot(list(testSpectra, testSpectraSmooth), mzRange = c(38.5, 40.5), type = 'l')
```

---

smootherSpline      *generic smootherSpline*

---

**Description**

generic smootherSpline  
method smootherSpline for TIC

**Usage**

```
smootherSpline(object, stepsize = 5, spar = 0.3, ...)

## S4 method for signature 'MassSpectra'
smootherSpline(object, stepsize = 5, spar = 0.3, ...)
```

**Arguments**

object	MassSpectra
stepsize	numeric arg for spline smoother
spar	numeric arg for spline smoother
...	additional args

**Value**

object of class MassSpectra with updated mass spectra  
object of class MassSpectra

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectraSmooth <- smootherSpline(testSpectra)
overlayPlot(list(testSpectra, testSpectraSmooth), mzRange = c(38.5, 40.5), type = 'l')
```

---

smoothScatter	<i>generic for smoothScatter</i>
---------------	----------------------------------

---

**Description**

smoothScatter method for PCA class

**Usage**

```
smoothScatter(  
  x,  
  y = NULL,  
  nbin = 128,  
  bandwidth,  
  colramp = colorRampPalette(c("white", blues9)),  
  nrpoints = 100,  
  ret.selection = FALSE,  
  pch = ".",  
  cex = 1,  
  col = "black",  
  transformation = function(x) x^0.25,  
  postPlotHook = box,  
  xlab = NULL,  
  ylab = NULL,  
  xlim,  
  ylim,  
  xaxs = par("xaxs"),  
  yaxs = par("yaxs"),  
  ...  
)
```

```
## S4 method for signature 'PCA'
```

```
smoothScatter(  
  x,  
  y = NULL,  
  nbin = 128,  
  bandwidth,  
  colramp = colorRampPalette(c("white", blues9)),  
  nrpoints = 100,  
  ret.selection = FALSE,  
  pch = ".",  
  cex = 1,  
  col = "black",  
  transformation = function(x) x^0.25,  
  postPlotHook = box,  
  xlab = NULL,  
  ylab = NULL,
```

```

xlim,
ylim,
xaxs = par("xaxs"),
yaxs = par("yaxs"),
...,
comps = c(1, 2),
pcType = "pcaScores",
label = FALSE,
labelThreshold = 1
)

```

### Arguments

x	object of class PCA
y	numeric usually NULL
nbin	numeric
bandwidth	numeric vector length 1 or 2
colramp	numeric
nrpoints	numeric
ret.selection	logical
pch	character
cex	numeric
col	character
transformation	function
postPlotHook	box
xlab	NULL
ylab	NULL
xlim	numeric
ylim	numeric
xaxs	par
yaxs	par
...	additional args
comps	numeric
pcType	character
label	boolean
labelThreshold	numeric

### Value

graphical output

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testImage<-PCAnalysis(testImage, nComp = 4)
smoothScatter(analysis(testImage, 1), comps = c(1,2),
pcType = 'pcaScores', xlab = 'comp 1', ylab = 'comp 2')
```

SNR

*Signal-to-Noise Ratio (SNR)***Description**

SNR function for MNF to calculate Signal to Noise Ratio

**Usage**

```
SNR(stat, x, y)
```

**Arguments**

stat	unknown
x	unknown
y	unknown

**Details**

function from mzimage to calculate signal-to-noise ratio function

**Value**

matrix numeric with signal-to-noise ratios

subset

*Generic method for subset***Description**

Generic method for subset

Subset method for objects of class MassImage

**Usage**

```
subset(x, ...)
```

```
## S4 method for signature 'MassImage'
```

```
subset(x, ..., xyUpperLeft = NULL, xyLowerRight = NULL)
```

**Arguments**

x                    object of class MassImage  
 ...                 additional args  
 xyUpperLeft        vector of length two with x and y for the upper left subset corner  
 xyLowerRight       vector of length two with x and y for the lower right subset corner

**Value**

object of class MassImage a subset of the in-object  
 object of class MassImage

**Examples**

```
library(tofsimsData)
data(tofsimsData)
subsetTestImage<-subset(testImage, xyUpperLeft = c(1,1), xyLowerRight = c(50,50))
image(subsetTestImage)
```

---

 unitMassPeaks

*Generic method for unitMassPeaks*


---

**Description**

Generic method for unitMassPeaks

**Usage**

```
unitMassPeaks(
  object,
  mzRange,
  widthAt,
  factor,
  upper = NULL,
  lower = NULL,
  ...
)

## S4 method for signature 'MassSpectra,numeric,numeric'
unitMassPeaks(
  object,
  mzRange,
  widthAt,
  factor,
  upper = NULL,
  lower = NULL,
  ...
)
```



**Arguments**

object	object of class MassSpectra
mzRange	vector numeric with lower and upper mass range limit for which to set unit mass peaks
widthAt	vector numeric two mass values at which to sample for peak width
factor	vector numeric two values summing up to 1 for setting assymmetric peak width limits
upper	vector numeric upper peak width limits
lower	vector numeric lower peak width limits
...	additional args

**Value**

object of class PeakList with unit mass peaks

**Examples**

```
library(tofsimsData)
data(tofsimsData)
testSpectra <- calibPointNew(testSpectra, mz = 15, value = 15.01551)
testSpectra <- calibPointNew(testSpectra, mz = 181, value = 181.0228)
testSpectra <- recalibrate(testSpectra)
testSpectra <- unitMassPeaks(testSpectra, mzRange = c(1,200), widthAt = c(15, 181),
factor = c(0.4, 0.6), lower = c(14.97, 15.05), upper = c(180.84, 181.43))
plot(testSpectra, mzRange = c(1,200), type = 'l')
```

---

validMassImageObject    *Validation method functionf for class MassImage objects*

---

**Description**

Validation method functionf for class MassImage objects

**Usage**

```
validMassImageObject(object)
```

**Arguments**

object	object of class MassImage
--------	---------------------------

**Value**

boolean class validity test

---

validMassSpectraObject

*Validation method function for class MassImage objects*

---

**Description**

Validation method function for class MassImage objects

**Usage**

validMassSpectraObject(object)

**Arguments**

object            object of class MassSpectra

**Value**

boolean class validity test

---

validPCAObject

*Validation method function for class PCA objects*

---

**Description**

Validation method function for class PCA objects

**Usage**

validPCAObject(object)

**Arguments**

object            object of class PCA

**Value**

boolean class validity test

---

validPeakListObject      *Validation method function for class PeakList objects*

---

**Description**

Validation method function for class PeakList objects

**Usage**

validPeakListObject(object)

**Arguments**

object                  object of class PeakList

**Value**

boolean class validity test

---

xdim                      *generic accessor method for "xdim"*

---

**Description**

generic accessor method for "xdim"

**Usage**

xdim(object)

**Arguments**

object                  object of class MassImage

**Value**

numeric value x dimension of mass image

---

xdim,MassImage-method *Getter, method definition "xdim" on "MassImage"*

---

**Description**

Getter, method definition "xdim" on "MassImage"

**Usage**

```
## S4 method for signature 'MassImage'  
xdim(object)
```

**Arguments**

object            objet of class MassImage

**Value**

numeric x dimension of slot xy

---

xdim,PCA-method            *method xdim() for PCA class object*

---

**Description**

method xdim() for PCA class object

**Usage**

```
## S4 method for signature 'PCA'  
xdim(object)
```

**Arguments**

object            object of class PCA

**Value**

numeric x dimension of image

---

xdim<-                    *generic setter method for "xdim"*

---

**Description**

generic setter method for "xdim"

**Usage**

```
xdim(object) <- value
```

**Arguments**

object	object of class MassImage
value	numeric x dimension of image

**Value**

object of class MassImage with updated x dimension

---

xy                            *xy, slot of MassImage class objects*

---

**Description**

xy, slot of MassImage class objects

**Usage**

```
xy(object)

xy(object) <- value

## S4 method for signature 'MassImage'
xy(object)

## S4 replacement method for signature 'MassImage'
xy(object) <- value
```

**Arguments**

object	object of class MassImage
value	vector numeric two values for x and y dimension of image

**Value**

vector numeric with xy dimensions of image

**Examples**

```
library(tofsimsData)
data(tofsimsData)
xy(testImage)
```

---

xySpec

*Generic method xySpec*

---

**Description**

Selection of Spectra

method xySpec extracts the mass spectra of positon x/y and puts them in a MassSpectar class object

**Usage**

```
xySpec(object, x = NULL, y = NULL)
```

```
## S4 method for signature 'MassImage'
xySpec(object, x = NULL, y = NULL)
```

**Arguments**

object	object of class MassImage
x	numeric x coordinate from where to sample a mass spectra
y	numeric y coordinate from where to sample a mass spectra

**Details**

Selection of mass spectra by vectors of equal length for x and y.

**Value**

object of class MassSpectra with selected mass spectra

**Author(s)**

Lorenz Gerber <lorenz.gerber@slu.se>

**Examples**

```
library(tofsimsData)
data(tofsimsData)
spectra100100<-xySpec(testImage, 100,100)
plot(spectra100100, type = 'l')
```

---

ydim                      *generic accessor method for "ydim"*

---

**Description**

generic accessor method for "ydim"

**Usage**

ydim(object)

**Arguments**

object                      object of class MassImage

**Value**

numeric integer, y dimension of image

---

ydim,MassImage-method    *Getter, method definition "ydim" on "MassImage"*

---

**Description**

Getter, method definition "ydim" on "MassImage"

**Usage**

```
## S4 method for signature 'MassImage'  
ydim(object)
```

**Arguments**

object                      object of class MassImage

**Value**

numeric y dimension of slot xy

---

ydim, PCA-method	<i>method ydim() for PCA class object</i>
------------------	---

---

**Description**

method ydim() for PCA class object

**Usage**

```
## S4 method for signature 'PCA'  
ydim(object)
```

**Arguments**

object            object of class PCA

**Value**

numeric y dimension of image

---

ydim<-	<i>generic setter method for "ydim"</i>
--------	---

---

**Description**

generic setter method for "ydim"

**Usage**

```
ydim(object) <- value
```

**Arguments**

object            object of class MassImage  
value             numeric y dimension of image

**Value**

updated object of type MassImage



---

zdim	<i>generic accessor method for "zdim"</i>
------	---

---

**Description**

generic accessor method for "zdim"

**Usage**

```
zdim(object)
```

**Arguments**

object            object of class MassImage

**Value**

numeric, number of mass channels / peaks

---

zdim,MassSpectra-method	<i>method definition 'zdim' on 'MassSpectra'</i>
-------------------------	--

---

**Description**

method definition 'zdim' on 'MassSpectra'

**Usage**

```
## S4 method for signature 'MassSpectra'  
zdim(object)
```

**Arguments**

object            object of class MassSpectra

**Value**

numeric value

# Index

- \* **package**
  - tofsims-package, 4
- addFixedWidth, 5
- addFixedWidth,PeakList,numeric,numeric-method (addFixedWidth), 5
- addPeaks, 6
- addPeaks,PeakList,missing,numeric-method (addPeaks), 6
- addPeaks,PeakList,numeric,numeric-method (addPeaks), 6
- analysis, 7, 8, 13, 14, 30, 48
- analysis,MassSpectra,missing-method (analysis), 7
- analysis,MassSpectra,numeric-method (analysis), 7
- analysis<- (analysis), 7
- analysis<- ,MassSpectra-method (analysis), 7
- analysisName, 7, 8, 13, 14, 30, 48
- analysisName,MassSpectra-method (analysisName), 8
- analysisName<- (analysisName), 8
- analysisName<- ,MassSpectra-method (analysisName), 8
  
- baseObject, 9
- baseObject,PrComp-method, 9
- baseObject,PrinComp-method, 10
- binning, 10
- binning,MassImage-method (binning), 10
- bwApply, 11
- bwApply,MassSpectra,matrix-method (bwApply), 11
  
- calibPointNew, 12
- calibPointNew,MassSpectra,numeric-method (calibPointNew), 12
- calibPoints, 7, 8, 13, 14, 30, 48
  
- calibPoints,MassSpectra-method (calibPoints), 13
- calibPoints<- (calibPoints), 13
- calibPoints<- ,MassSpectra-method (calibPoints), 13
- calibration, 7, 8, 13, 14, 30, 48
- calibration,MassSpectra-method (calibration), 14
- calibration<- (calibration), 14
- calibration<- ,MassSpectra-method (calibration), 14
- changePeakWidth, 15
- changePeakWidth,PeakList,missing,missing,missing-method (changePeakWidth), 15
- changePeakWidth,PeakList,numeric,numeric,numeric-method (changePeakWidth), 15
- check.extension, 16
- computeMNF, 17
- computeNoise, 18
- coordToPixel, 18
- coordToPixel,MassImage,numeric-method, 19
- covDiffCalc, 19
- cReadRawPhi, 20
- cTable (ctable), 21
- ctable, 21
  
- dim,MassImage-method, 21
- dim,MassSpectra-method, 22
  
- EigenDecompose, 22
- extract.header.data, 23
  
- findClosestMatch, 23
- findPeakWidth, 24
- findPeakWidth,PeakList-method (findPeakWidth), 24
  
- getTOFs, 25
- getTOFs,MassSpectra-method (getTOFs), 25

- image, 26
- image, MassImage-method (image), 26
- image, PCA-method (image), 26
- imageMatrix, 27
- imageMatrix, MassImage-method (imageMatrix), 27
- imageMatrix, PCA-method (imageMatrix), 27
- import, 28
- import.raw, 28
- instrument, 7, 8, 13, 14, 29, 48
- instrument, MassSpectra-method (instrument), 29
- instrument<- (instrument), 29
- instrument<-, MassSpectra-method (instrument), 29
- iters, 30
- iters, MCR-method, 31
- itzipName, 31
- itzipName<-, 32
  
- LapackGenEigen, 32
- legend.col, 33
- look.for.itzip.property, 33
  
- MAF, 34
- makeTIC, 35
- makeTIC, MassSpectra-method, 35
- manualSelectPeaks, 36
- MassImage, 36
- MassSpectra, 7, 8, 13, 14, 30, 38, 48
- MCR (MCR-class), 39
- MCR-class, 39
- MNF, 40
- mz, 7, 8, 13, 14, 30, 48
- mz, MassSpectra-method, 41
- mz<- , MassSpectra-method (mz, MassSpectra-method), 41
  
- nComp, 42
- nComp, PCA-method (nComp), 42
- ndim, 43
- ndim, MassSpectra-method, 43
- nearestNeighbourMean, 44
- nnMean, 44
- nnMNF, 45
- noPlottingData, 46
- noPlottingData, PCA-method, 46
- nPeaks, 47
- nPeaks, PeakList-method (nPeaks), 47
  
- nz, 7, 8, 13, 14, 30, 47
- nz, MassSpectra, missing-method (nz), 47
- nz, MassSpectra, numeric-method (nz), 47
- nz<- (nz), 47
- nz<- , MassSpectra-method (nz), 47
  
- opaMCR (MCR-class), 39
- overlayPlot, 48
- overlayPlot, list-method (overlayPlot), 48
  
- parIndicesSearch, 49
- PCA (PCA-class), 50
- PCA-class, 50
- pcaLoadings, 51
- pcaLoadings, PCA, missing-method (pcaLoadings), 51
- pcaLoadings, PCA, numeric-method (pcaLoadings), 51
- pcaMAF, 52
- PCAnalysis, 52
- pcaScores, 53
- pcaScores, PCA, ANY-method (pcaScores), 53
- pcaScores, PCA, numeric-method (pcaScores), 53
- peakIDs, 54
- peakIDs, PeakList-method (peakIDs), 54
- peakIDs<- (peakIDs), 54
- peakIDs<- , PeakList-method (peakIDs), 54
- PeakList, 55
- peakMzs, 56
- peakMzs, PeakList-method (peakMzs), 56
- peakMzs<- (peakMzs), 56
- peakMzs<- , PeakList-method (peakMzs), 56
- peakPick, 57
- peakPick, MassSpectra-method (peakPick), 57
- peaks2Spectra, 58
- peaks2Spectra, PeakList, MassSpectra-method (peaks2Spectra), 58
- peakWidths, 59
- peakWidths, PeakList-method (peakWidths), 59
- plot, 60
- plot, MassImage, missing-method, 61
- plot, MassSpectra, missing-method (plot), 60
- plot, PCA, ANY-method (plot), 60
- plot, PeakList, missing-method, 62

- points, [63](#)
- points,MassSpectra-method (points), [63](#)
- poissonScaling, [64](#)
- poissonScaling,MassSpectra-method (poissonScaling), [64](#)
- PrComp (PrComp-class), [65](#)
- prComp (PrComp-class), [65](#)
- PrComp-class, [65](#)
- PrinComp (PrinComp-class), [66](#)
- prinComp (PrinComp-class), [66](#)
- PrinComp-class, [66](#)
  
- readBIF, [67](#)
- recalibrate, [68](#)
- recalibrate,MassSpectra-method (recalibrate), [68](#)
- reduceSpectrumResolution, [68](#)
- reduceSpectrumResolution,MassSpectra-method (reduceSpectrumResolution), [68](#)
- removePeaks, [69](#)
- removePeaks,PeakList,missing,character,numeric,missing-method (removePeaks), [69](#)
- removePeaks,PeakList,missing,missing,missing,missing-method (removePeaks), [69](#)
- removePeaks,PeakList,missing,missing,numeric,missing-method (removePeaks), [69](#)
- removePeaks,PeakList,numeric,missing,missing,missing-method (removePeaks), [69](#)
- resids, [71](#)
- resids,MCR-method, [71](#)
- RSS, [72](#)
- RSS,MCR-method, [72](#)
  
- scale, [73](#)
- scale,MassSpectra-method (scale), [73](#)
- show,MassImage-method, [74](#)
- show,MassSpectra-method, [74](#)
- show,PeakList-method, [75](#)
- smootherGolay, [75](#)
- smootherGolay,MassSpectra-method (smootherGolay), [75](#)
- smootherSpline, [76](#)
- smootherSpline,MassSpectra-method (smootherSpline), [76](#)
- smoothScatter, [77](#)
- smoothScatter,PCA-method (smoothScatter), [77](#)
- SNR, [79](#)
- subset, [79](#)
- subset,MassImage-method (subset), [79](#)
- tofsims-package, [4](#)
- unitMassPeaks, [80](#)
- unitMassPeaks,MassSpectra,numeric,numeric-method (unitMassPeaks), [80](#)
  
- validMassImageObject, [81](#)
- validMassSpectraObject, [82](#)
- validPCAObject, [82](#)
- validPeakListObject, [83](#)
  
- xdim, [83](#)
- xdim,MassImage-method, [84](#)
- xdim,PCA-method, [84](#)
- xdim<-, [85](#)
- xy, [85](#)
- xy,MassImage-method (xy), [85](#)
- xy<- (xy), [85](#)
- xy<-,MassImage-method (xy), [85](#)
- xySpec,missing-method (xySpec), [86](#)
- xySpec,MassImage-method (xySpec), [86](#)
- ydim, [87](#)
- ydim,MassImage-method (ydim), [87](#)
- ydim,PCA-method, [88](#)
- ydim<-, [88](#)
- zdim, [89](#)
- zdim,MassSpectra-method, [89](#)