

# Package: MALDIcellassay (via r-universe)

October 22, 2024

**Type** Package

**Title** Automated MALDI Cell Assays Using Dose-Response Curve Fitting

**Version** 0.4.47.9000

**Description** Conduct automated cell-based assays using Matrix-Assisted Laser Desorption/Ionization (MALDI) methods for high-throughput screening of signals responsive to treatments. The package efficiently identifies high variance signals and fits dose-response curves to them. Quality metrics such as  $Z'$ ,  $V'$ , log2 Fold-Change, and Curve response score (CRS) are provided for evaluating the potential of signals as biomarkers. The methodologies were introduced by Weigt et al. (2018) <[doi:10.1038/s41598-018-29677-z](https://doi.org/10.1038/s41598-018-29677-z)> and refined by Unger et al. (2021) <[doi:10.1038/s41596-021-00624-z](https://doi.org/10.1038/s41596-021-00624-z)>.

**License** MIT + file LICENSE

**Imports** methods, ggplot2, nplr, dplyr, tidyr, forcats, scales, MALDIquant, MALDIquantForeign, tibble, svMisc, purrr

**Encoding** UTF-8

**LazyData** true

**LazyDataCompression** xz

**RoxygenNote** 7.3.2

**Roxygen** list(markdown = TRUE)

**Suggests** rmarkdown, knitr, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Depends** R (>= 4.2)

**URL** <https://github.com/CeMOS-Mannheim/MALDIcellassay>,  
<https://cemos-mannheim.github.io/MALDIcellassay/>

**BugReports** <https://github.com/CeMOS-Mannheim/MALDIcellassay/issues>

**Config/testthat/edition** 3

**Repository** <https://cemos-mannheim.r-universe.dev>

**RemoteUrl** <https://github.com/cemos-mannheim/maldicellassay>

**RemoteRef** HEAD

**RemoteSha** 4d6438215552bd0f586c1b0505c168bac9d9d868

## Contents

Blank2022intmat . . . . .	3
Blank2022peaks . . . . .	4
Blank2022res . . . . .	4
Blank2022spec . . . . .	5
calculateChauvenetCriterion . . . . .	6
calculateCurveFit . . . . .	7
calculateSSMD . . . . .	7
calculateVPrime . . . . .	8
calculateZPrime . . . . .	9
checkRecalibration . . . . .	10
extractIntensity . . . . .	11
extractSpots . . . . .	12
filterVariance . . . . .	12
fitCurve . . . . .	13
getAllMz . . . . .	14
getAppliedMzShift . . . . .	15
getAppliedNormFactors . . . . .	16
getAvgPeaks . . . . .	16
getAvgSpectra . . . . .	17
getBinTol . . . . .	17
getConc . . . . .	18
getCurveFits . . . . .	18
getFittingParameters . . . . .	19
getIntensityMatrix . . . . .	19
getMzFromMzIdx . . . . .	20
getMzShift . . . . .	21
getNormFactors . . . . .	21
getNormMethod . . . . .	22
getNormMz . . . . .	23
getNormMzTol . . . . .	23
getPeakStatistics . . . . .	24
getRecalibrationError . . . . .	24
getSinglePeaks . . . . .	25
getSingleSpecIntensity . . . . .	25
getSNR . . . . .	26
getSpots . . . . .	26
getVarFilterMethod . . . . .	27
isMALDIassay . . . . .	28
loadSpectra . . . . .	28
loadSpectraMzML . . . . .	29
MALDIassay-class . . . . .	30
normalize . . . . .	30

<i>Blank2022intmat</i>	3
normalizeByFactor . . . . .	31
peaks2df . . . . .	31
plotCurves . . . . .	32
plotPeak . . . . .	33
shiftMassAxis . . . . .	33
transformConc2Log . . . . .	34
<b>Index</b>	<b>35</b>

---

Blank2022intmat	<i>Example intensity matrix from Blank 2022</i>
-----------------	---

---

## Description

Intensity matrix from MALDI mass spectrometry data of EOC cells treated with different concentrations of SAHA. It is used to demonstrate the usage of MALDIcell assay.

## Usage

```
data(Blank2022intmat)
```

## Format

Matrix with concentrations of original spectra as rownames and m/z-values as colnames.

## Details

The concentrations include: 0, 0.04, 0.12, 0.37, 1.11, 3.33, 10 and 30 uM of SAHA at 4 replicates each. The original spectra were trimmed to 400-900 Da mass-range to keep the file size small. The peaks are the result of `MALDIquant::intensityMatrix(Blank2022peaks, Blank2022spec)`

## References

Blank, M., Enzlein, T. & Hopf, C. LPS-induced lipid alterations in microglia revealed by MALDI mass spectrometry-based cell fingerprinting in neuroinflammation studies. *Sci Rep* 12, 2908 (2022). <https://doi.org/10.1038/s41598-022-06894-1>

Blank2022peaks

*Example peaks from Blank 2022*

---

**Description**

Peaks from MALDI mass spectrometry data of EOC cells treated with different concentrations of SAHA. It is used to demonstrate the usage of MALDIcellassay.

**Usage**

```
data(Blank2022peaks)
```

**Format**

A list of MALDIquant::MassPeaks-objects named with the respective concentration.

**Details**

The concentrations include: 0, 0.04, 0.12, 0.37, 1.11, 3.33, 10 and 30 uM of SAHA at 4 replicates each. The original spectra were trimmed to 400-900 Da mass-range to keep the file size small. The peaks are the result of applying MALDIquant::detectPeaks to Blank2022spec with arguments SNR = 3, method = "SuperSmoother".

**References**

Blank, M., Enzlein, T. & Hopf, C. LPS-induced lipid alterations in microglia revealed by MALDI mass spectrometry-based cell fingerprinting in neuroinflammation studies. *Sci Rep* 12, 2908 (2022). <https://doi.org/10.1038/s41598-022-06894-1>

---

Blank2022res

*Example MALDIassay object from Blank 2022*

---

**Description**

Object of class MALDIcellassay from MALDI mass spectrometry data of EOC cells treated with different concentrations of SAHA. It is used to demonstrate the usage of MALDIcellassay.

**Usage**

```
data(Blank2022res)
```

**Format**

Matrix with concentrations of original spectra as rownames and m/z-values as colnames.

**Details**

The concentrations include: 0, 0.04, 0.12, 0.37, 1.11, 3.33, 10 and 30 uM of SAHA at 4 replicates each. The original spectra were trimmed to 400-900 Da mass-range to keep the file size small. The peaks are the result of `fitCurve(spec = Blank2022spec, SinglePointRecal = TRUE, normMz = 760.585, alignTol = 0.1, normTol = 0.1)`

**References**

Blank, M., Enzlein, T. & Hopf, C. LPS-induced lipid alterations in microglia revealed by MALDI mass spectrometry-based cell fingerprinting in neuroinflammation studies. *Sci Rep* 12, 2908 (2022). <https://doi.org/10.1038/s41598-022-06894-1>

---

Blank2022spec

*Example spectra from Blank 2022*

---

**Description**

MALDI mass spectrometry data of EOC cells treated with different concentrations of SAHA. It is used to demonstrate the usage of MALDIcell assay.

**Usage**

```
data(Blank2022spec)
```

**Format**

A list of MALDIquant::MassSpectrum-objects named with the respective concentration.

**Details**

The concentrations include: 0, 0.04, 0.12, 0.37, 1.11, 3.33, 10 and 30 uM of SAHA at 4 replicates each. The original spectra were trimmed to 400-900 Da mass-range to keep the file size small.

**References**

Blank, M., Enzlein, T. & Hopf, C. LPS-induced lipid alterations in microglia revealed by MALDI mass spectrometry-based cell fingerprinting in neuroinflammation studies. *Sci Rep* 12, 2908 (2022). <https://doi.org/10.1038/s41598-022-06894-1>

---

`calculateChauvenetCriterion`*Calculate Chauvenet's criterion for outlier detection*

---

## Description

Calculate Chauvenet's criterion for outlier detection

## Usage

```
calculateChauvenetCriterion(x)
```

## Arguments

`x` numeric, values (e.g. intensities) to test for outliers

## Details

Note that, as for all outlier detection criteria: Excluding data points from your measurement should only be conducted with extreme care. Even if this (or any other) function tells you that a data point is an outlier, you might still want to have it in your sample population especially if you are not sure if your data is normal distributed. See [Wikipedia](#) for details of the algorithm.

## Value

logical vector, TRUE for detected outliers.

## Examples

```
set.seed(42)

#no outlier
sample <- rnorm(n = 8, mean = 0, sd = 0.01)
calculateChauvenetCriterion(sample)

# introduce outlier
sample[1] <- 1
calculateChauvenetCriterion(sample)
```

---

calculateCurveFit      *Calculate the fit for a dose-response curve*

---

**Description**

Calculate the fit for a dose-response curve

**Usage**

```
calculateCurveFit(intmat, idx, verbose = TRUE, ...)
```

**Arguments**

intmat	Intensity matrix as generated by MALDIquant::intensityMatrix() with row-names as the respective concentrations of the spectra.
idx	Numeric vector of the mz indices to perform the fit.
verbose	Logical, print logs to console.
...	Additional arguments passed to nplr::nplr().

**Value**

List of curve fits.

**Examples**

```
data(Blank2022intmat)

# for faster runtime we let it run on 5 peaks only
fits <- calculateCurveFit(Blank2022intmat, idx = 1:5)
```

---

calculateSSMD      *Calculate strictly standardized mean difference (SSMD)*

---

**Description**

Calculate strictly standardized mean difference (SSMD)

**Usage**

```
calculateSSMD(res, internal = TRUE, nConc = 2)
```

**Arguments**

res	Object of class MALDIassay
internal	Logical, currently only the internal implementation, using nConc top and bottom concentrations, is implemented.
nConc	Numeric, number of top and bottom concentrations to be used to calculate the pseudo positive and negative control. Only used if internal is TRUE

**Details**

The strictly standardized mean difference (SSMD) is a measure of effect size. It is the mean divided by the standard deviation of a difference between the positive and negative control.

$$\gamma = \frac{|\mu_n - \mu_p|}{\sqrt{\sigma_n^2 + \sigma_p^2}}$$

The SSMD can be easily be interpreted as it denotes the difference between positive and negative controls in units of standard deviation.

**Value**

Numeric vector of strictly standardized mean differences (SSMD)

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)

calculateSSMD(Blank2022res, nConc = 2)
```

---

calculateVPrime

*Calculate V'-Factor*

---

**Description**

Calculate V'-Factor

**Usage**

```
calculateVPrime(res, internal = TRUE)
```

**Arguments**

res	Object of class MALDIassay
internal	Logical, currently only the internal implementation, using nConc top and bottom concentrations, is implemented.



**Details**

The V'-factor is a generalization of the Z'-factor to a dose-response curve. See [M.-A. Bray and A. Carpenter, Advanced assay development guidelines for image-based high content screening and analysis](#) for details. It is defined as:

$$V' = 1 - 6 * \sigma_f / |\mu_p - \mu_n|$$

with

$$\sigma_f = \sqrt{1/N * \sum y_{fit} - y_{measured}^2}$$

In other words,  $\sigma_f$  is the standard deviation of residuals.

Note, we do not need to estimate the variance for the mean of the positive and negative value. So, this function uses the top and bottom asymptote directly instead of taking the top and bottom concentrations in consideration.

**Value**

Numeric vector of V'-factors

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)

calculateVPrime(Blank2022res)
```

---

calculateZPrime	<i>Calculate Z'-factor of assay quality</i>
-----------------	---

---

**Description**

Calculate Z'-factor of assay quality

**Usage**

```
calculateZPrime(res, internal = TRUE, nConc = 2)
```

**Arguments**

res	Object of class MALDIassay
internal	Logical, currently only the internal implementation, using nConc top and bottom concentrations, is implemented.
nConc	Numeric, number of top and bottom concentrations to be used to calculate the pseudo positive and negative control. Only used if internal is TRUE

## Details

The most common way to measure the quality of an assay is the so-called  $Z'$ -factor, which describes the separation of the positive and negative control in terms of their standard deviations  $\sigma_p$  and  $\sigma_n$ . The  $Z'$ -factor is defined as [Ji-Hu Zhang et al., A simple statistical parameter for use in evaluation and validation of high throughput screening assays.](#)

$$Z' = 1 - (3 * (\sigma_p + \sigma_n)) / |\mu_p - \mu_n|$$

where  $\mu_p$  and  $\mu_n$  is the mean value of the positive (response expected) and negative (no response expected) control, respectively. Therefore, the assay quality is **independent of the shape of the concentration response curve** and solely depend on two control values.

Note, if internal is set to TRUE, the nConc highest concentrations are assumed as positive control, whereas the nConc lowest concentrations are used as negative.

Value	Interpretation
$Z' \sim 1$	perfect assay
$1 > Z' > 0.5$	excellent assay
$0.5 > Z' > 0$	moderate assay
$Z' = 0$	good only for yes/no response
$Z' < 0$	unacceptable

## Value

Numeric vector of  $Z'$ -factors.

## Examples

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
calculateZPrime(Blank2022res, nConc = 2)
```

---

checkRecalibration      *Check the recalibration of spectra from a MALDI assay object*

---

## Description

Dashed gray lines indicate the m/z used for re-calibration  $\pm$  the tolerance. Red dashed line indicate the m/z used for re-calibration and solid lines indicate peaks. The spectrum will show the peak used for re-calibration  $\pm$  10x the tolerance.

## Usage

```
checkRecalibration(object, idx)
```

**Arguments**

object            Object of class MALDIassay  
idx                Numeric, index of spectrum to plot

**Value**

ggplot object

**Examples**

```
# see example for `fitCurve()` to see how this data was generated  
data(Blank2022res)  
checkRecalibration(Blank2022res, idx = 1:8)
```

---

extractIntensity            *Extract intensity using peaks as template*

---

**Description**

Extract intensity using peaks as template

**Usage**

```
extractIntensity(mz, peaks, spec, tol)
```

**Arguments**

mz                numeric, m/z values to be extracted from the peaks/spectra  
peaks             MALDIquant::MassPeaks list  
spec              MALDIquant::MassSpectrum list  
tol                numeric, tolerance in Da

**Value**

MALDIquant::MassPeaks list with extracted intensities from spec at m/z of peaks = pseudo peaks.  
Useful in combination with sdMassSpectrum to get standard deviation of peaks as intensity matrix.

**Examples**

```
data(Blank2022peaks)  
data(Blank2022spec)  
  
int <- extractIntensity(mz = c(409, 423, 440),  
                      peaks = Blank2022peaks,  
                      spec = Blank2022spec,  
                      tol = 0.2)  
  
head(int)
```

---

extractSpots	<i>Extract the spot coordinates</i>
--------------	-------------------------------------

---

**Description**

Extract the spot coordinates

**Usage**

```
extractSpots(spec)
```

**Arguments**

spec            list of MALDIquant::MassSpectrum or MALDIquant::MassPeaks objects

**Value**

Character vector of spot names. If multiple spots are used (e.g. for average spectra) they will be concatenate.

**Examples**

```
data(Blank2022spec)
head(extractSpots(Blank2022spec))
```

---

filterVariance	<i>Filter for high variance signals</i>
----------------	---

---

**Description**

Filter for high variance signals

**Usage**

```
filterVariance(
  vars,
  method = c("mean", "median", "q25", "q75", "none"),
  verbose = TRUE
)
```

**Arguments**

vars            Numeric vector, variances of signals

method          Character, filtering method. One of "mean" (default), "median", "q25", "q75" (25 and 75% quantile) or "none".

verbose         Logical, print logs to console.

**Value**

Indices of spectra with a high variance

**Examples**

```
data(Blank2022intmat)
# get variance of each peak
vars <- apply(Blank2022intmat, 2, var)
highVarIndices <- filterVariance(vars, method = "mean", verbose = TRUE)
```

---

fitCurve

*Fit dose-response curves*


---

**Description**

Fit dose-response curves

**Usage**

```
fitCurve(
  spec,
  unit = c("M", "mM", "uM", "nM", "pM", "fM"),
  varFilterMethod = c("mean", "median", "q25", "q75", "none"),
  monoisotopicFilter = FALSE,
  averageMethod = c("mean", "median", "sum"),
  normMz = NULL,
  normTol = 0.1,
  alignTol = 0.01,
  binTol = 2e-04,
  SNR = 3,
  halfWindowSize = 3,
  allowNoMatches = TRUE,
  normMeth = c("mz", "TIC", "PQN", "median", "none"),
  SinglePointRecal = TRUE,
  verbose = TRUE
)
```

**Arguments**

spec	List of MALDIquant::MassSpectrum
unit	Character, unit of concentration. Used to calculate the concentration in Moles so that pIC50 is correct. Set to "M" if you dont want changes in your concentrations.
varFilterMethod	Character, function applied for high variance filtering. One of the following options mean (default), median, q25, q75 or none (no filtering).

monoisotopicFilter	Logical, filter peaks and just use monoisotopic peaks for curve fit.
averageMethod	Character, aggregation method for average mass spectra ("mean" or "median")
normMz	Numeric, m/z used for normalization AND for single point recalibration.
normTol	Numeric, tolerance in Dalton to match normMz
alignTol	Numeric, tolerance for spectral alignment in Dalton.
binTol	Numeric, tolerance for binning of peaks.
SNR	Numeric, signal to noise ratio for peak detection.
halfWindowSize	2ction. See MALDIquant::detectPeaks().
allowNoMatches	Logical, if normMz can not be found in a spectrum, proceed and exclude spectrum or stop
normMeth	Character, normalization method. Can either be "TIC", "PQM", "median" or "mz". If "mz" then the normMz is used. If none no normalization is done.
SinglePointRecal	Logical, perform single point recalibration to normMz
verbose	Logical, print logs to console.

**Value**

Object of class MALDIassay. The most important slot is fits which contains the IC50 curve fits.

**Examples**

```
data(Blank2022spec)

fitCurve(spec = Blank2022spec,
         SinglePointRecal = TRUE,
         normMz = 760.585,
         alignTol = 0.1,
         normTol = 0.1,
         varFilterMethod = "mean")
```

---

getAllMz

*Get all m/z value of an MALDIassay-object*

---

**Description**

Get all m/z value of an MALDIassay-object

**Usage**

```
getAllMz(object, excludeNormMz = FALSE)
```

**Arguments**

object            Object of class MALDIassay  
excludeNormMz   Logical, remove normMz from list of mz values.

**Value**

numeric vector of mz values

**Examples**

```
# see example for `fitCurve()` to see how this data was generated  
data(Blank2022res)  
head(getAllMz(Blank2022res))
```

---

*getAppliedMzShift*        *Extract applied mz-shift*

---

**Description**

Extract applied mz-shift

**Usage**

```
getAppliedMzShift(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric vector of mz-shifts applied to spectra

**Examples**

```
# see example for `fitCurve()` to see how this data was generated  
data(Blank2022res)  
head(getAppliedMzShift(Blank2022res))
```

---

getAppliedNormFactors *Extract applied normalization factors*

---

**Description**

Extract applied normalization factors

**Usage**

```
getAppliedNormFactors(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric vector of normalization factors applied to spectra

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getAppliedNormFactors(Blank2022res))
```

---

getAvgPeaks            *Extract peaks of average spectra*

---

**Description**

Extract peaks of average spectra

**Usage**

```
getAvgPeaks(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

List of MALDIquantMassPeaks

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getAvgPeaks(Blank2022res)[[1]]
```



---

getAvgSpectra	<i>Extract average spectra</i>
---------------	--------------------------------

---

**Description**

Extract average spectra

**Usage**

```
getAvgSpectra(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

List of MALDIquantMassSpectrum

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getAvgSpectra(Blank2022res)[[1]]
```

---

getBinTol	<i>Get binning tolerance</i>
-----------	------------------------------

---

**Description**

Get binning tolerance

**Usage**

```
getBinTol(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric, tolerance used for binning in Dalton.

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getBinTol(Blank2022res)
```

---

getConc	<i>Extract the concentrations used in a MALDIassay</i>
---------	--

---

**Description**

Extract the concentrations used in a MALDIassay

**Usage**

```
getConc(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric vector, concentrations used in a MALDIassay

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getConc(Blank2022res))
```

---

getCurveFits	<i>Extract curve fits</i>
--------------	---------------------------

---

**Description**

Extract curve fits

**Usage**

```
getCurveFits(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

List, containing the data used to do the fits as well as the nlpr curve fit .

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
fits <- getCurveFits(Blank2022res)
```

---

getFittingParameters *Get fitting parameters*

---

**Description**

Get fitting parameters

**Usage**

```
getFittingParameters(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

tibble of fitting parameters for each fitted m/z-value

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getFittingParameters(Blank2022res))
```

---

getIntensityMatrix *Get the intensity matrix of single spectra for all fitted curves*

---

**Description**

Get the intensity matrix of single spectra for all fitted curves

**Usage**

```
getIntensityMatrix(object, avg = FALSE, excludeNormMz = FALSE)
```

**Arguments**

object            Object of class MALDIassay

avg               Logical, return single spectra intensity matrix (default) or average spectra intensity matrix

excludeNormMz    Logical, exclude normMz from intensity matrix.

### Details

Note that the returned matrix only contains  $m/z$  values that were actually fitted. If a variance filtering step was applied this will not include **all**  $m/z$  values. If you wish to get a matrix of **all**  $m/z$  values use `MALDIquant::intensityMatrix(getSinglePeaks(object))`. For average spectra intensity matrix with **all**  $m/z$  values use `MALDIquant::intensityMatrix(getAvgPeaks(object), getAvgSpectra(object))`.

### Value

A matrix with columns as  $m/z$  values and rows as concentrations/spectra

### Examples

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getIntensityMatrix(Blank2022res, avg = TRUE, excludeNormMz = TRUE) )
```

---

getMzFromMzIdx	<i>Get the mz value associated with a mzIdx</i>
----------------	---

---

### Description

Get the mz value associated with a mzIdx

### Usage

```
getMzFromMzIdx(object, mzIdx)
```

### Arguments

object	Object of class MALDIassay
mzIdx	numeric, index of mass of interest (see <code>getPeakStatistics()</code> )

### Value

numeric, mz value

### Examples

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getMzFromMzIdx(Blank2022res, mzIdx = 2)
```

---

getMzShift                      *Get mass shift for target mz*

---

**Description**

Get mass shift for target mz

**Usage**

```
getMzShift(peaks, targetMz, tol, tolppm = FALSE, verbose = TRUE)
```

**Arguments**

peaks	List of MALDIquant::MassPeak
targetMz	Numeric, target mass
tol	Numeric, tolerance around targetMz
tolppm	Logical, tolerance supplied in ppm
verbose	Logical, print logs to the console.

**Value**

List with two entries: `MzShift` The mass shift for each spectrum `specIdx` The index of the spectra with a match for `targetMz`

**Examples**

```
data(Blank2022peaks)
getMzShift(Blank2022peaks, targetMz = 760.585, tol = 0.1, tolppm = FALSE)
```

---

getNormFactors                      *Get normalization factors from peak data.frame*

---

**Description**

Get normalization factors from peak data.frame

**Usage**

```
getNormFactors(peaksdf, targetMz, tol, tolppm = TRUE, allowNoMatch = TRUE)
```

**Arguments**

peaksdf	data.frame with peaks information as generated by peaks2df()
targetMz	Numeric, target mass
tol	Numeric, tolerance around targetMz
tolppm	Logical, is the tolerance provided in ppm (TRUE) or Dalton (FALSE)
allowNoMatch	Logical, stop if targetMz is not found in single spectrum? If TRUE spectra without targetMz match will be excluded.

**Value**

List with two entries:

norm_factor	The normalization factor for each spectrum
specIdx	The index of the spectra with a match for targetMz

**Examples**

```
data(Blank2022peaks)
getNormFactors(peaks2df(Blank2022peaks), targetMz = 760.585, tol = 0.1, tolppm = FALSE)
```

---

getNormMethod	<i>Extract normalization method</i>
---------------	-------------------------------------

---

**Description**

Extract normalization method

**Usage**

```
getNormMethod(object)
```

**Arguments**

object	Object of class MALDIassay
--------	----------------------------

**Value**

Character, normalization method used.

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getNormMethod(Blank2022res)
```

---

getNormMz	<i>Extract m/z used for normalization</i>
-----------	---

---

**Description**

Extract m/z used for normalization

**Usage**

```
getNormMz(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric, m/z used for normalization

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getNormMz(Blank2022res)
```

---

getNormMzTol	<i>Extract tolerance used for normalization</i>
--------------	---

---

**Description**

Extract tolerance used for normalization

**Usage**

```
getNormMzTol(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric, tolerance used for normalization

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getNormMzTol(Blank2022res)
```

---

getPeakStatistics      *Extract peak statistics*

---

**Description**

Extract peak statistics

**Usage**

```
getPeakStatistics(object, summarise = FALSE)
```

**Arguments**

object	Object of class MALDIassay
summarise	Logical, return summarized results (one result per m/z and not per m/z and spectra)

**Value**

A tibble with peak statistics ( $R^2$ , fold-change, CV%, etc.)

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getPeakStatistics(Blank2022res, summarise = TRUE))
```

---

getRecalibrationError      *Calculate remaining calibration error of a MALDIassay object*

---

**Description**

Calculate remaining calibration error of a MALDIassay object

**Usage**

```
getRecalibrationError(object)
```

**Arguments**

object	Object of class MALDIassay
--------	----------------------------

**Value**

A tibble containing statistics about remaining calibration error



**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getRecalibrationError(Blank2022res)
```

---

getSinglePeaks	<i>Extract peaks of single spectra spectra (before average calculation)</i>
----------------	---

---

**Description**

Extract peaks of single spectra spectra (before average calculation)

**Usage**

```
getSinglePeaks(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

List of MALDIquantMassPeaks

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getSinglePeaks(Blank2022res)[[1]]
```

---

getSingleSpecIntensity	<i>Extract the intensities of single spectra for a given mzIdx</i>
------------------------	--

---

**Description**

Extract the intensities of single spectra for a given mzIdx

**Usage**

```
getSingleSpecIntensity(object, mz_idx)
```

**Arguments**

object            Object of class MALDIassay  
mz\_idx            Integer, index of mz

**Value**

Numeric vector, intensities of mzIdx

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
head(getSingleSpecIntensity(Blank2022res, 2))
```

---

getSNR	<i>Extract SNR used for peak detection</i>
--------	--

---

**Description**

Extract SNR used for peak detection

**Usage**

```
getSNR(object)
```

**Arguments**

object            Object of class MALDIassay

**Value**

Numeric, SNR used for peak detection

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getSNR(Blank2022res)
```

---

getSpots	<i>Get the spot coordinates of spectra</i>
----------	--

---

**Description**

Get the spot coordinates of spectra

**Usage**

```
getSpots(object, singleSpec = TRUE)
```

**Arguments**

object	Object of class MALDIassay
singleSpec	Logical, extract the spot coordinates of single spectra (default) or from average spectra.

**Value**

character vector of spot coordinates. In case of average spectra multiple spots are concatenated.

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
# spots per spectrum
getSpots(Blank2022res, singleSpec = TRUE)

#spots per concentration
getSpots(Blank2022res, singleSpec = FALSE)
```

---

getVarFilterMethod     *Extract variance filtering method*

---

**Description**

Extract variance filtering method

**Usage**

```
getVarFilterMethod(object)
```

**Arguments**

object	Object of class MALDIassay
--------	----------------------------

**Value**

Character of variance filtering method used

**Examples**

```
# see example for `fitCurve()` to see how this data was generated
data(Blank2022res)
getVarFilterMethod(Blank2022res)
```

---

isMALDIassay	<i>Check if object is of class MALDIassay</i>
--------------	---

---

**Description**

Check if object is of class MALDIassay

**Usage**

```
isMALDIassay(object)
```

**Arguments**

object	object to text
--------	----------------

**Value**

logical, TRUE if object is of class MALDIassay

**Examples**

```
x <- 1
# FALSE
isMALDIassay(x)
# TRUE
isMALDIassay(Blank2022res)
```

---

loadSpectra	<i>load bruker MALDI target plate spectra</i>
-------------	---

---

**Description**

load bruker MALDI target plate spectra

**Usage**

```
loadSpectra(Dir, filter = NA, nameSpectra = TRUE, verbose = TRUE)
```

**Arguments**

Dir	Character, parent directory of spectra.
filter	Character vector, filter out spectra which match the given vector.
nameSpectra	Logical, if TRUE the spectra in the resulting list will be named according to the dirname.
verbose	Logical, print logs to the console.

**Value**

List of MALDIquant::MassSpectra

**Examples**

```
dataDir <- system.file("extdata", package="MALDIcellassay")
unzip(file.path(dataDir, "example-raw-spectra.zip"))

loadSpectra("example-raw-spectra/")

unlink("example-raw-spectra/", recursive = TRUE)
```

---

loadSpectraMzML	<i>load mzML spectra</i>
-----------------	--------------------------

---

**Description**

load mzML spectra

**Usage**

```
loadSpectraMzML(Dir, filter = NA, nameSpectra = TRUE, verbose = TRUE)
```

**Arguments**

Dir	Character, parent directory of spectra.
filter	Character vector, filter out spectra which match the given vector.
nameSpectra	Logical, if TRUE the spectra in the resulting list will be named according to the dirname.
verbose	Logical, print logs to console

**Value**

List of MALDIquant::MassSpectra

**Examples**

```
dataDir <- system.file("extdata", package="MALDIcellassay")

loadSpectraMzML(file.path(dataDir, "Koch2024mzML"))
```

---

MALDIassay-class	<i>Class MALDIassay</i>
------------------	-------------------------

---

**Description**

A class for holding MALDI assay related information.

**Arguments**

object	MALDIassay.
--------	-------------

---

normalize	<i>Normalize spectra and peaks</i>
-----------	------------------------------------

---

**Description**

Normalize spectra and peaks

**Usage**

```
normalize(spec, peaks, normMeth, normMz, normTol)
```

**Arguments**

spec	List of MALDIquant::MassSpectrum
peaks	List of MALDIquant::MassPeaks
normMeth	Character, normalization method. Options are "TIC", "median" and "mz".
normMz	Numeric, mz used to normalize.
normTol	Numeric, tolerance around normMz.

**Value**

List of lists of normalized MALDIquant::MassSpectrum, normalized MALDIquant::MassPeaks, normalization factors as well as indices of spectra containing the normMz in case of normMeth = "mz",

**Examples**

```
data(Blank2022spec)
data(Blank2022peaks)
norm <- normalize(Blank2022spec, Blank2022peaks, normMeth = "mz", normMz = 760.585, normTol = 0.1)

# normalization factors
norm$factor
```

---

normalizeByFactor      *Apply normalization factors to spectra*

---

**Description**

Apply normalization factors to spectra

**Usage**

```
normalizeByFactor(spec, factors)
```

**Arguments**

spec                    List of MALDIquant::MassSpectrum or MALDIquant::MassPeaks  
factors                Numeric vector of normalization factors. See getNormFactors().

**Value**

List of normalized Spectra or Peaks

**Examples**

```
#' data(Blank2022peaks)
normFactors <- getNormFactors(peaks2df(Blank2022peaks),
                              targetMz = 760.585,
                              tol = 0.1,
                              tolppm = FALSE)
normPeaks <- normalizeByFactor(Blank2022peaks,
                              normFactors$norm_factor)
```

---

peaks2df                *Convert a list of peaks to a data.frame*

---

**Description**

Convert a list of peaks to a data.frame

**Usage**

```
peaks2df(peaks)
```

**Arguments**

peaks                    (list of) MALDIquant::MassPeaks

**Value**

Data.frame with peak data

**Examples**

```
data(Blank2022peaks)

peakdf <- peaks2df(Blank2022peaks[1:2])
head(peakdf)
```

---

plotCurves	<i>generate ggplot objects for each of the curve fits in a MALDIassay object</i>
------------	--

---

**Description**

generate ggplot objects for each of the curve fits in a MALDIassay object

**Usage**

```
plotCurves(object, mzIdx = NULL, errorbars = c("none", "sd", "sem"))
```

**Arguments**

object	object of class MALDIassay
mzIdx	numeric, indices of m/z values to plot (see <code>getPeakStatistics()</code> ). Note, <code>fc_thresh</code> and <code>R2_thresh</code> filters do not apply if <code>mzIdx</code> is set!
errorbars	character, add error bars to plot. Either standard error of the mean ( <code>sem</code> ) or standard deviation ( <code>sd</code> ) in regards to the measurement replicates or no errorbars ( <code>none</code> ).

**Value**

list of ggplot objects

**Examples**

```
data(Blank2022res)
plotCurves(Blank2022res, mzIdx = 2, errorbars = "sd")
```



---

plotPeak	<i>Plot a peak of interest from a MALDIassay object</i>
----------	---

---

**Description**

Plot a peak of interest from a MALDIassay object

**Usage**

```
plotPeak(object, mzIdx, tol = 0.8)
```

**Arguments**

object	object of class MALDIassay
mzIdx	numeric, index of mass of interest (see <code>getPeakStatistics()</code> )
tol	numeric, tolerance around peak to plot

**Value**

ggplot object

**Examples**

```
data(Blank2022res)
plotPeak(Blank2022res, mzIdx = 2)
```

---

shiftMassAxis	<i>Shift mass axis</i>
---------------	------------------------

---

**Description**

Shift mass axis

**Usage**

```
shiftMassAxis(spec, mzdiff)
```

**Arguments**

spec	List of MALDIquant::MassSpectrum or MALDIquant::MassPeaks
mzdiff	Numeric vector, see <code>getMzShift()</code>

**Value**

List of MALDIquant::MassSpectrum or MALDIquant::MassPeaks with shifted mass axis.

**Examples**

```
data(Blank2022spec)
# raw mz
head(Blank2022spec[[1]]@mass)

# shifted mz
shifted <-shiftMassAxis(Blank2022spec[1:2], c(0.5, 0.5))
head(shifted[[1]]@mass)
```

---

transformConc2Log      *Convert concentration to log10 and replace zero's*

---

**Description**

Convert concentration to log10 and replace zero's

**Usage**

```
transformConc2Log(conc)
```

**Arguments**

conc                  numeric, concentrations.

**Value**

numeric, log10 transformed concentrations

**Examples**

```
transformConc2Log(c(0.1, 0.01, 0.001))
```

# Index

## \* datasets

- Blank2022intmat, [3](#)
  - Blank2022peaks, [4](#)
  - Blank2022res, [4](#)
  - Blank2022spec, [5](#)
- Blank2022intmat, [3](#)  
Blank2022peaks, [4](#)  
Blank2022res, [4](#)  
Blank2022spec, [5](#)
- calculateChauvenetCriterion, [6](#)  
calculateCurveFit, [7](#)  
calculateSSMD, [7](#)  
calculateVPrime, [8](#)  
calculateZPrime, [9](#)  
checkRecalibration, [10](#)
- extractIntensity, [11](#)  
extractSpots, [12](#)
- filterVariance, [12](#)  
fitCurve, [13](#)
- getAllMz, [14](#)  
getAppliedMzShift, [15](#)  
getAppliedNormFactors, [16](#)  
getAvgPeaks, [16](#)  
getAvgSpectra, [17](#)  
getBinTol, [17](#)  
getConc, [18](#)  
getCurveFits, [18](#)  
getFittingParameters, [19](#)  
getIntensityMatrix, [19](#)  
getMzFromMzIdx, [20](#)  
getMzShift, [21](#)  
getNormFactors, [21](#)  
getNormMethod, [22](#)  
getNormMz, [23](#)  
getNormMzTol, [23](#)  
getPeakStatistics, [24](#)
- getRecalibrationError, [24](#)  
getSinglePeaks, [25](#)  
getSingleSpecIntensity, [25](#)  
getSNR, [26](#)  
getSpots, [26](#)  
getVarFilterMethod, [27](#)
- isMALDIassay, [28](#)
- loadSpectra, [28](#)  
loadSpectraMzML, [29](#)
- MALDIassay-class, [30](#)
- normalize, [30](#)  
normalizeByFactor, [31](#)
- peaks2df, [31](#)  
plotCurves, [32](#)  
plotPeak, [33](#)
- shiftMassAxis, [33](#)
- transformConc2Log, [34](#)