# Package 'CorrectOverloadedPeaks'

May 7, 2026

**Type** Package

**Title** Correct Overloaded Peaks from GC-APCI-MS Data

**Version** 1.3.5

**Date** 2025-02-24

**Maintainer** Jan Lisec <jan.lisec@bam.de>

**Description** Analyzes and modifies metabolomics raw data (generated using Gas Chromatography-Atmospheric Pressure Chemical Ionization-Mass Spectrometry) to correct overloaded signals, i.e. ion intensities exceeding detector saturation leading to a cut-off peak. Data in 'xcmsRaw' format are accepted as input and 'mzXML' files can be processed alternatively. Overloaded signals are detected automatically and modified using an Gaussian or an Isotopic-Ratio approach. Quality control plots are generated and corrected data are stored within the original 'xcmsRaw' or 'mzXML' respectively to allow further processing.

**License** GPL-3

**URL** https://github.com/janlisec/CorrectOverloadedPeaks

**BugReports** https://github.com/janlisec/CorrectOverloadedPeaks/issues

**VignetteBuilder** knitr

**Encoding** UTF-8

**Depends** R(>= 2.10.0)

**Imports** methods

**Suggests** bitops, digest, knitr, rmarkdown, spelling, xcms, XML, xml2

**RoxygenNote** 7.3.2

**Language** en-US

**NeedsCompilation** no

**Author** Jan Lisec [aut, cre] (ORCID: <https://orcid.org/0000-0003-1220-2286>)

**Repository** CRAN

**Date/Publication** 2025-02-27 00:50:05 UTC

# Contents



---

`base64decode` *base64decode.*

---

## Description

'base64decode' is a copy of a similar function from the caTools package as this package is about to be archived (07/2018).

## Usage

```
base64decode(z, what, size = NA, signed = TRUE, endian = .Platform$endian)

base64encode(x, size = NA, endian = .Platform$endian)
```

## Arguments

| | |
|---|---|
| `z` | The base64 encoded string. |
| `what` | Define output type of z (e.g. 'numeric'). |
| `size` | Encoding size (provide if you know it). |
| `signed` | Parameter passed through to 'readBin'. |
| `endian` | Parameter passed through to 'readBin'. |
| `x` | The value vector to be encoded. |

## Details

'base64decode' will convert base64 encoded strings into R values.#'

## Value

Decoded value of z.#'

## Examples

```
## Not run:
# you need to have the bitops package installed to run the example
x <- c(10, 0.2, 123456)
(z <- base64encode(x = x))
base64decode(z = z, what = "numeric")
(x <- as.integer(x))
(z <- base64encode(x = x))
base64decode(z = z, what = "int")

## End(Not run)
```

---

`CorrectOverloadedPeaks` *Correct Overloaded Peaks from GC-MS data.*

---

## Description

`CorrectOverloadedPeaks` will take an xcmsRaw data structure (or any imported mzXML) and search for overloaded peaks within the mass traces. It will correct overloaded peaks automatically using an Gaussian or IsotopicRatio approach, generate QC plots and write the corrected data back into the original xcmsRaw.

## Usage

```
CorrectOverloadedPeaks(
  data = NULL,
  method = c("Isoratio", "Gauss", "EMG"),
  detection_limit = 1,
  ds = NULL,
  silent = TRUE,
  testing = FALSE,
  attotwm = FALSE,
  region = NULL,
  peak = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | An xcmsRaw-object or an mzXML-object as imported by `read.mzXML`. |
| `method` | Either Gauss or EMG (usually better results) or Isoratio (more robust for non-Gaussian peak shapes). |
| `detection_limit` | If=1 only peaks hitting detector saturation (ds) will be corrected, can be lowered to 0.95 to catch also peaks going into saturation. |
| `ds` | Detector saturation. Will be determined based on data if not specified explicitly. |

| | |
|---|---|
| silent | QC-plots will be generated if silent=FALSE and additional Warnings() will be generated. |
| testing | Will automatically set silent=FALSE and store all extracted regions with overloaded peaks in the working directory as cor_df_all.RData. |
| attotwm | All-the-Time-of-the-World-Mode. If calculation time doesn't matter try this out. :) |
| region | From an initial QC-Plot file you may reprocess a specific overloaded region. Don't forget to specify the ds parameter explicitly. |
| peak | You may further restrict the reprocessing to a specific peak within the region. |

## Details

This is a high level function to batch pre-process metabolomics data which are partially overloaded before continuing with the standard workflow of peak identification etc.. It relies internally on FitGaussPeak and FitPeakByIsotopicRatio to modify data of individual intensity signal. Basically the function aims to identify automatically overloaded regions and extracts base peak chromatograms for all overloaded m/z traces within these regions, which are corrected and put back into the original data structure. For simplicity some potentially interesting parameters are hidden at the top of the function definition. They have been set to values determined empirically to be working for a Bruker impact II MS (high-res QTOF) coupled to GC and LC via APCI and ESI respectively. For more details please see doi:10.1021/acs.analchem.6b02515.

## Value

An corrected xcmsRaw- or mzXML-object which can be exported to file. Additionally a QC-plot pdf-file if silent=FALSE.

## References

doi:10.1021/acs.analchem.6b02515

## See Also

ModelGaussPeak

FitGaussPeak

FitPeakByIsotopicRatio

read.mzXML

write.mzXML

## Examples

```
## Not run:
  # load mzXML test data
  data(mzXML_data)
  CorrectOverloadedPeaks(data = mzXML_data, method = "EMG", silent = FALSE)

## End(Not run)
```

FitGaussPeak *Extrapolate a flat top peak using a Gauss approach.*

## Description

FitGaussPeak will take retention time ('x') and intensity ('y') data and extrapolate all points above a certain threshold based on further parameters using a Gaussian approach.

## Usage

```
FitGaussPeak(
  x,
  y,
  scale_range = c(1, 10),
  steps = 10,
  cutoff = 0.95,
  idx = NULL,
  weight_front = 0.5,
  strip_data = "none",
  account_for_baseline_offset = TRUE,
  method = c("Gauss", "EMG")[1],
  silent = TRUE,
  fix_sd = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| x | A numeric vector, retention times. |
| y | A numeric vector, ion intensities. |
| scale_range | Specifies the expected range for the true peak to exceed the observed, where scale_range=c(1,100) would assume anything between not overloaded and 100-fold overloaded. |
| steps | Specifies a step parameter used to create a sequence within 'scale_range' to test for good fits, higher=more precision, fewer=faster. |
| cutoff | Overloaded peaks will be screwed from Gaussian shape already when approaching detector saturation (DS), cutoff=0.95 ensures that points just before DS will not be used to model fit. |
| idx | If not NULL, 'idx' is expected to specify points to correct explicitly (as a numeric-vector within 1:length(x)). |
| weight_front | A weighting parameter to punish deviations in peak front and tail differently; 0.5=use front/tail equally, 1=use only front, 0=use only tail. |
| strip_data | Use all provided data if 'none' (default). Strip 'front' or 'tail' data in case you observe peak fronting or tailing respectively. |

account_for_baseline_offset
: If TRUE will subtract min(y) from y before fitting parameters.

method
: The method for peak shape calculation. Can be 'Gauss' or 'EMG' (exponentially modified gauss).

silent
: For testing purposes some QC-plot will be generated if silent=FALSE.

fix_sd
: Supply a fix standard deviation (sd) for the peak or leave NULL to estimate sd within function.

...
: passed to the QC plot function, e.g. 'main' or 'xlab'.

## Details

This function is mainly used internally (CorrectOverloadedPeaks) but can be of value on it's own to test brute force peak reconstruction given that appropriate base peak chromatograms are available.

## Value

An annotated plot of the mass spectrum and detailed information within the console (if silent=FALSE) and the optimal fitted data points (vector of length(y), returned invisible).

## Examples

```
#load test data
data("mzXML_data")
names(mzXML_data)
str(mzXML_data[["scan"]][[1]])
pk <- ModelGaussPeak(height=10^7, width=3, scan_rate=10, e=0, ds=8*10^6, base_line=10^2)
plot(pk, main="Gaussian peak of true intensity 10^7 but cutt off at 8*10^6")
idx <- pk[,"int"]>0.005 * max(pk[,"int"])
tmp <- FitGaussPeak(x=pk[idx,"rt"], y=pk[idx,"int"], silent=FALSE, xlab="RT", ylab="Intensity")
```

---

# FitPeakByIsotopicRatio

*Extrapolate a flat top peak using isotopic ratios.*

---

## Description

FitPeakByIsotopicRatio will take a data frame containing peak data for retention time ('RT'), as well as mass and intensity information of M0, M+1 and M+2 and extrapolate all points above a certain threshold for Int_M0 based on further parameters using an IsotopicRatio approach.

## Usage

```
FitPeakByIsotopicRatio(cor_df = NULL, idx = NULL, silent = TRUE)
```

### Arguments

| | |
|---|---|
| cor_df | A data frame containing information about the overloaded area; columns=(Scan, RT, mz0, int0, mz1, int1, mz2, int2, modified). |
| idx | If not NULL, 'idx' is expected to specify points to correct explicitly (as a numeric-vector within 1:length(x)). |
| silent | For testing purposes some QC-plot will be generated if silent=FALSE. |

### Details

Isotopic ratios within ion traces of molecules can be considered stable. If this ratio is changed because one molecule, let's say the M+0, is exceeding the detector range while another (say M+1) is still quantifiable, we therefore may attempt to modify M+0 by multiplying the values of M+1 with a constant (the stable ratio). This constant is determined ideally from the values within the peak front. As this function is mainly used internally (CorrectOverloadedPeaks), it is not very flexible with respect to the input format. Please prepare a dataframe according to the parameter specifications or process a file using CorrectOverloadedPeaks with testing=TRUE, which will generate a list structure of such dataframes.

### Value

An annotated plot of the mass spectrum and detailed information within the console. Main result will be returned invisible.

---

## ModelGaussPeak *Create and modify parameters of an artificial chromatographic peak.*

---

### Description

ModelGaussPeak will create a potentially overloaded Gaussian peak of requested width and height.

### Usage

```
ModelGaussPeak(
  height = 10^7,
  width = 4,
  scan_rate = 10,
  e = 0,
  ds = 10^7,
  base_line = 10^2
)
```

### Arguments

| | |
|---|---|
| height | True peak height (=intensity counts). |
| width | Peak width in time units (preferably seconds). |
| scan_rate | Is determining the resolution of data points per time unit (preferably seconds). |

| | |
|---|---|
| e | Error term giving the percent amount of deviation from the ideal Gaussian curve for individual data points. |
| ds | Detector saturation. Intensity values will be cut off at this point if requested. |
| base_line | Defines if peak is supposed to have a higher base level. |

## Details

The main task of ModelGaussPeak is to create peak data in Gaussian shape for testing. Width is meant in the chromatographic sense, i.e. the time between peak front and tail hitting the baseline.

## Value

Dataframe with columns 'rt' and 'int'.

## Examples

```
ylim <- c(0,10^7)
par(mfrow=c(1,5))
pk <- ModelGaussPeak(height=10^7, width=4, scan_rate=10, e=0, ds=10^7, base_line=10^2)
plot(pk,ylim=ylim,main="standard")
pk <- ModelGaussPeak(height=10^7, width=4, scan_rate=10, e=0, ds=8*10^6, base_line=10^2)
plot(pk,ylim=ylim,main="flat top")
pk <- ModelGaussPeak(height=10^7, width=4, scan_rate=10, e=0, ds=8*10^6, base_line=10^5)
plot(pk,ylim=ylim,main="high baseline")
pk <- ModelGaussPeak(height=10^7, width=4, scan_rate=10, e=0.05, ds=8*10^6, base_line=10^5)
plot(pk,ylim=ylim,main="e=5%")
pk <- ModelGaussPeak(height=10^7, width=4, scan_rate=5, e=0.05, ds=8*10^6, base_line=10^5)
plot(pk,ylim=ylim,main="sr=5")
```

---

# mzXML_data *mzXML_data.*

---

## Description

mzXML_data.

## Usage

```
data(mzXML_data)
```

## Format

A object of class mzXML. A test dataset imported by read.mzXML() from a GC-APCI measurement on a Bruker impact II, exported by Compass to an mzXML file.

## Source

Jan Lisec (jan.lisec@bam.de)

`read.mzData` *read.mzData.*

## Description

`read.mzData` will import mzData as xcmsRaw-class objects.

## Usage

```
read.mzData(filename, fmt = c("xcmsRaw", "xcmsRawLike"), verbose = FALSE)
```

## Arguments

| | |
|---|---|
| `filename` | A path to a mzData file (as exported by 'xcms::write.mzdata()'). |
| `fmt` | Output format. Currently 'xcmsRaw' and 'xcmsRawLike' are supported. The latter is an S4 class similar to xcmsRaw but allowing to omit the xcms package. |
| `verbose` | Print messages to console. |

## Details

The main task of read.mzData functions is to import mzData files to R. Currently 'xcmsRaw' is supported as an output format. I created this function for legacy reasons as the mzData import is no longer supported by 'mzR' and consequently 'xcms' since 09/2021. This is a quick and dirty implementation. It will work only for mslevel=1 and a fixed set of base64 encoding parameters (size = 4, endian = "big"). However, feel free to send me an e-mail if you are interested in using the function but cant get it working.

## Value

A generic R object of class xcmsRaw.

## Examples

```
## Not run:
  data(mzXML_data)
  write.mzXML(mzXML = mzXML_data, filename = "test.mzXML")
  x <- xcms::xcmsRaw("test.mzXML",  profstep=0)
  xcms::write.mzdata(x, file="test.mzData")
  x2 <- read.mzData(filename = "test.mzData")
  identical(str(x), str(x2))
  identical(x@env$intensity, x2@env$intensity)
  identical(x@env$mz, x2@env$mz)
  identical(x@scanindex, x2@scanindex)
  file.remove(c('test.mzData', 'test.mzXML'))

  # check that objects are independent (not identical)
  identical(methods::new("xcmsRawLike"), methods::new("xcmsRawLike"))
```

```
## End(Not run)
```

---

`read.mzXML` *Read and write standard mzXML files.*

---

## Description

'read.mzXML' and 'write.mzXML' are copied from the caMassClass package which is no longer actively on CRAN.

## Usage

```
read.mzXML(filename)

write.mzXML(mzXML, filename, precision = c("32", "64"))

new.mzXML()
```

## Arguments

| | |
|---|---|
| `filename` | The mzXML file name to be read or written. |
| `mzXML` | The generic mzXML object |
| `precision` | Either '32' or '64' byte. |

## Details

The main task of 'read.mzXML' and 'write.mzXML' functions is to extract and save scan data of mzXML files. In addition attempt is made to keep all other sections of mzXML file as unparsed XML code, so the data can be extracted latter or saved into new mzXML files. Those unparsed sections are stored as XML text.

## Value

A generic R object of class mzXML for 'read.mzXML' and NULL for 'write.mzXML'.

Nothing. mzXML object is exported to filename. If it was imported using `read.mzXML` it should contain all previous fields.

# Index