Package: squallms (via r-universe)

October 6, 2026

Type Package

Title Speedy quality assurance via lasso labeling for LC-MS data

Version 1.7.0

Description squallms is a Bioconductor R package that implements a ``semi-labeled'' approach to untargeted mass spectrometry data. It pulls in raw data from mass-spec files to calculate several metrics that are then used to label MS features in bulk as high or low quality. These metrics of peak quality are then passed to a simple logistic model that produces a fully-labeled dataset suitable for downstream analysis.

License MIT + file LICENSE

URL https://github.com/wkumler/squallms

BugReports https://github.com/wkumler/squallms/issues

Encoding UTF-8

RoxygenNote 7.3.1

biocViews MassSpectrometry, Metabolomics, Proteomics, Lipidomics, ShinyApps, Classification, Clustering, FeatureExtraction, PrincipalComponent, Regression, Preprocessing, QualityControl, Visualization

Depends R (>= 3.5.0)

Imports xcms, MSnbase, MsExperiment, RaMS, dplyr, tidyr, tibble, ggplot2, shiny, plotly, data.table, caret, stats, graphics, utils, keys

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

VignetteBuilder knitr

Config/testthat/edition 3

Config/pak/sysreqs
cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libnetcdf-dev libssl-dev zlib1g-dev

Repository https://bioc.r-universe.dev

Date/Publication 2026-04-28 13:03:18 UTC

RemoteUrl https://github.com/bioc/squallms

RemoteRef HEAD

RemoteSha 454253cb0112ddd0576a789bfd024230084fd8b2

Contents

extractChromMetrics 2
labelFeatsLasso 4
labelFeatsManual 5
logModelFeatProb 6
logModelFeatQuality 8
makeXcmsObjFlat 9
pickyPCA 10
updateXcmsObjFeats 11

Index **14**

`extractChromMetrics` *Extract metrics of chromatographic peak quality*

Description

This function takes flat-form XC-MS data (i.e. the format produced by 'makeXcmsObjectFlat') and calculates metrics of peak shape and similarity for downstream processing (e.g. by `updateXcmsObjFeats`). The core metrics are those described in https://doi.org/10.1186/s12859-023-05533-4, corresponding to peak shape (correlation coefficient between the raw data and an idealized bell curve, beta_cor) and within-peak signal-to-noise (maximum intensity divided by the standard deviation of the residuals after the best-fitting bell is subtracted from the raw data). Additionally, the function interpolates the raw data to fixed retention time intervals and performs a principal components analysis (PCA) across a file-by-RT matrix which typically extracts good-looking peaks in the first component (PC1). These functions are calculated on the raw MS data as obtained via **RaMS**, so either the filepaths must be included in the peak_data object or supplied as ms1_data.

Usage

```
extractChromMetrics(
  peak_data,
  recalc_betas = FALSE,
  ms1_data = NULL,
  verbosity = 0
)
```

Arguments

peak_data	Flat-form XC-MS data with columns for the bounding box of a chromatographic peak (mzmin, mzmax, rtmin, rtmax) as grouped by a feature ID. Must be provided WITHOUT retention time correction for proper matching to the values in the raw data.
recalc_betas	Scalar boolean controlling whether existing beta values (as calculated in the XCMS object when peakpicked with `CentWaveParam(verboseBetaColumns=TRUE)`) should be used as-is (the default) or recalculated. See https://github.com/sneumann/xcms/pull/685 for differences in these implementations.
ms1_data	Optional data.table object produced by RaMS containing MS1 data with columns for filename, rt, mz, and int. If not provided, the files are detected from the filepath column in peak_data.
verbosity	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

A data.frame containing one row for each feature in peak_data, with columns containing the median peak shape value (med_cor), and the median SNR value (med_snr).

Examples

```
library(xcms)
library(MSnbase)
library(dplyr)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
feat_metrics <- extractChromMetrics(peak_data, recalc_betas = TRUE)
```

`labelFeatsLasso` *Label similar chromatographic features in bulk via interactive selection*

Description

This function interpolates multi-file chromatograms to a shared set of retention time points then performs a PCA to place similar chromatograms near each other in a reduced dimensionality space. Features can then be labeled in groups instead of one at a time, massively reducing the burden of creating a high-quality dataset. This implementation relies on R's **shiny** package to provide interactive support in a browser environment and the **plotly** package for selection tools. Classified features are then returned as a simple R object for downstream use.

Usage

```
labelFeatsLasso(
  peak_data,
  ms1_data = NULL,
  rt_window_width = 1,
  ppm_window_width = 5,
  verbosity = 1
)
```

Arguments

`peak_data`	Flat-form XC-MS data with columns for the bounding box of a chromatographic peak (mzmin, mzmax, rtmin, rtmax) as grouped by a feature ID. Must be provided WITHOUT retention time correction for proper matching to the values in the raw data.
`ms1_data`	Optional data.table object produced by RaMS containing MS1 data with columns for filename, rt, mz, and int. If not provided, the files are detected from the filepath column in peak_data.
`rt_window_width`	The width of the retention time window that should be used for PCA construction, in minutes.
`ppm_window_width`	The width of the m/z window that should be used for PCA construction, in parts per million.
`verbosity`	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

A character vector named with feature IDs containing the classifications of each peak that was viewed during the interactive phase. NA values indicate those features that were not classified.

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
if (interactive()) {
    lasso_labels <- labelFeatsLasso(peak_data)
}
```

`labelFeatsManual` *Label chromatographic features manually one at a time via interactive interface*

Description

This function allows the user to view and label individual chromatographic features using the keyboard. Running labelFeatsManual launches a browser which shows a single feature extracted from multiple files. The feature can then be classified by pressing a key (defaults are left=bad, right=good) on the keyboard which is recorded and a new feature is automatically shown.This implementation relies on R's **shiny** package to provide interactive support in a browser environment and the **keys** package for key binding within a browser. Classified features are then returned as a simple R object for downstream use.

Usage

```
labelFeatsManual(peak_data, ms1_data = NULL, verbosity = 1)
```

Arguments

`peak_data`	Flat-form XC-MS data with columns for the bounding box of a chromatographic peak (mzmin, mzmax, rtmin, rtmax) as grouped by a feature ID. Must be provided WITHOUT retention time correction for proper matching to the values in the raw data.

ms1_data	Optional data.table object produced by RaMS containing MS1 data with columns for filename, rt, mz, and int. If not provided, the files are detected from the filepath column in peak_data.
verbosity	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing while 1 will report diagnostic messages.

Value

A character vector named with feature IDs containing the classifications of each peak that was viewed during the interactive phase. NA values indicate those features that were not classified.

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
if (interactive()) {
    manual_labels <- labelFeatsManual(peak_data)
}
```

---

logModelFeatProb *Model feature quality using a logistic regression*

Description

After metrics have been extracted (typically with ‘extractChromMetrics‘) and labeling has occurred (typically with ‘labelFeatsManual‘ or ‘labelFeatsLasso‘), a model can be built to robustly classify the peaks that were not labeled based on the metrics. The default regression equation fits the med_cor and med_snr columns in feature_metrics to the feature_labels, but additional metrics can be passed as well (but be careful of overfitting!).

Usage

```
logModelFeatProb(
  feature_metrics,
  feature_labels,
  log_formula = feat_class ~ med_cor + med_snr,
  verbosity = 2
)
```

Arguments

feature_metrics	A data.frame with columns used to construct the feature quality model.
feature_labels	A character vector named with feature IDs and entries corresponding to the peak quality (either "Good", "Bad", or NA).
log_formula	The formula to use when predicting feature quality from the feature metrics. This formula is passed to ‘glm‘ as-is, so make sure that the predictive features exist in the feature_metrics data.frame.
verbosity	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

A numeric vector of probabilities returned by the logistic model named by feature ID

Examples

```
library(xcms)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") |>
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files |>
    readMSData(msLevel. = 1, mode = "onDisk") |>
    findChromPeaks(cwp) |>
    adjustRtime(obp) |>
    groupChromPeaks(pdp) |>
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
feat_metrics <- extractChromMetrics(peak_data, verbosity = 0)

# Load demo labels previously assigned using the lasso method
lasso_classes <- readRDS(system.file("extdata", "intro_lasso_labels.rds", package = "squallms"))
feat_probs <- logModelFeatProb(feat_metrics, lasso_classes)
```

`logModelFeatQuality` *Turn 0-1 likelihood values into categorical (good/bad) classifications*

Description

This function wraps 'logModelFeatProb' for modeling of feature quality to estimate the quality of every feature in the dataset and classify them as good/bad based on whether they exceed the provided 'likelihood_threshold'. Lower likelihood thresholds (0.01, 0.1) will produce more false positives (noise peaks included when they shouldn't be). Higher likelihood thresholds (0.9, 0.99) will produce more false negatives (good peaks removed when they shouldn't be).

Usage

```
logModelFeatQuality(
  feature_metrics,
  feature_labels,
  log_formula = feat_class ~ med_cor + med_snr,
  likelihood_threshold = 0.5,
  verbosity = 2
)
```

Arguments

`feature_metrics`	A data.frame with columns used to construct the feature quality model.
`feature_labels`	A character vector named with feature IDs and entries corresponding to the peak quality (either "Good", "Bad", or NA).
`log_formula`	The formula to use when predicting feature quality from the feature metrics. This formula is passed to 'glm' as-is, so make sure that the predictive features exist in the feature_metrics data.frame.
`likelihood_threshold`	A scalar numeric above which features will be kept if their predicted probability exceeds.
`verbosity`	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

A character vector of feature quality assessments returned by the logistic model and named by feature ID

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
```

```
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
feat_metrics <- extractChromMetrics(peak_data, verbosity = 0)

# Load demo labels previously assigned using the lasso method
lasso_classes <- readRDS(system.file("extdata", "intro_lasso_labels.rds", package = "squallms"))
feat_classes <- logModelFeatQuality(feat_metrics, lasso_classes)
```

`makeXcmsObjFlat` *Make an XCMS object flat*

Description

XCMSnExp objects are complicated S4 objects that make it difficult to access information about the features and their associated peaks. This function turns the output into a "flat" file format (a data.frame) so it can interact with tidyverse functions more easily.

Usage

```
makeXcmsObjFlat(xcms_obj, revert_rts = TRUE, verbosity = 0)
```

Arguments

`xcms_obj`	A XCMSnExp object produced by the typical XCMS workflow which should include retention time correction and peak correspondence and filling
`revert_rts`	Scalar boolean controlling whether the adjusted retention times found in the XCMS object are propagated or returned as-is
`verbosity`	Scalar numberic. Will report XCMS info if greater than zero or run silently if not (the default).

Value

A data.frame with columns for feature information (from featureDefinitions: feature, feat_mzmed, feat_rtmed, feat_npeaks, and peakidx) and peak information (from chromPeaks: mz, mzmin, mzmax, rt, rtmin, rtmax, into, intb, maxo, sn, sample) as well as the full path to the associated file and the file name alone (filepath and filename).

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
makeXcmsObjFlat(xcms_filled)
```

---

pickyPCA *Perform a PCA on multi-file chromatographic data*

Description

Internal function, mostly.

Usage

```
pickyPCA(
  peak_data,
  ms1_data,
  rt_window_width = NULL,
  ppm_window_width = NULL,
  verbosity = 1
)
```

Arguments

peak_data	Flat-form XC-MS data with columns for the bounding box of a chromatographic peak (mzmin, mzmax, rtmin, rtmax) as grouped by a feature ID. Must be provided WITHOUT retention time correction for proper matching to the values in the raw data.
ms1_data	Optional data.table object produced by RaMS containing MS1 data with columns for filename, rt, mz, and int. If not provided, the files are detected from the filepath column in peak_data.

rt_window_width
: The width of the retention time window that should be used for PCA construction, in minutes.

ppm_window_width
: The width of the m/z window that should be used for PCA construction, in parts per million.

verbosity
: Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

A list with two named components, interp_df and pcamat. interp_df is the MS1 data associated with each peak interpolated to a retention time spacing shared across the feature. pcamat is the result of cleaning this object up, pivoting it wider, and converting it to a matrix so that a PCA can be performed.

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
msdata <- RaMS::grabMSdata(unique(peak_data$filepath), grab_what = "MS1", verbosity = 0)
pixel_pca <- pickyPCA(peak_data, msdata$MS1)
```

---

updateXcmsObjFeats *Update features in an XCMS object*

Description

After metrics have been extracted (typically with 'extractChromMetrics') and labeling has occurred (typically with 'labelFeatsManual' or 'labelFeatsLasso'), low quality features can be removed from the XCMS object to improve downstream processing. This function wraps 'logModelFeatQuality' and automatically edits the features within the provided XCMS object.

Usage

```
updateXcmsObjFeats(
  xcms_obj,
  feature_metrics,
  feature_labels,
  log_formula = feat_class ~ med_cor + med_snr,
  likelihood_threshold = 0.5,
  verbosity = 2
)
```

Arguments

xcms_obj	The XCMS object from which features were initially extracted, usually with 'extractChromMetrics'
feature_metrics	A data.frame with columns used to construct the feature quality model.
feature_labels	A character vector named with feature IDs and entries corresponding to the peak quality (either "Good", "Bad", or NA).
log_formula	The formula to use when predicting feature quality from the feature metrics. This formula is passed to 'glm' as-is, so make sure that the predictive features exist in the feature_metrics data.frame.
likelihood_threshold	A scalar numeric above which features will be kept if their predicted probability exceeds.
verbosity	Scalar value between zero and two determining how much diagnostic information is produced. 0 should return nothing, 1 should return text-based progress markers, and 2 will return diagnostic plots if available.

Value

The initial xcms_obj but only containing features that exceed the provided quality threshold (as established in likelihood space)

Examples

```
library(xcms)
library(dplyr)
library(MSnbase)
mzML_files <- system.file("extdata", package = "RaMS") %>%
    list.files(full.names = TRUE, pattern = "[A-F].mzML")
register(BPPARAM = SerialParam())
cwp <- CentWaveParam(snthresh = 0, extendLengthMSW = TRUE, integrate = 2)
obp <- ObiwarpParam(binSize = 0.1, response = 1, distFun = "cor_opt")
pdp <- PeakDensityParam(
    sampleGroups = 1:3, bw = 12, minFraction = 0,
    binSize = 0.001, minSamples = 0
)
xcms_filled <- mzML_files %>%
    readMSData(msLevel. = 1, mode = "onDisk") %>%
```

```
    findChromPeaks(cwp) %>%
    adjustRtime(obp) %>%
    groupChromPeaks(pdp) %>%
    fillChromPeaks(FillChromPeaksParam(ppm = 5))
peak_data <- makeXcmsObjFlat(xcms_filled)
feat_metrics <- extractChromMetrics(peak_data, verbosity = 0)
lasso_classes <- readRDS(system.file("extdata", "intro_lasso_labels.rds", package = "squallms"))
xcms_filled <- updateXcmsObjFeats(xcms_filled, feat_metrics, lasso_classes)
```

Index

extractChromMetrics, 2

labelFeatsLasso, 4
labelFeatsManual, 5
logModelFeatProb, 6
logModelFeatQuality, 8

makeXcmsObjFlat, 9

pickyPCA, 10

updateXcmsObjFeats, *2*, 11