Package ‘Cardinal’

May 1, 2024

Type Package
Title A mass spectrometry imaging toolbox for statistical analysis
Version 3.6.0
Date 2021-11-21
Description Implements statistical & computational tools for analyzing mass spectrometry imaging datasets, including methods for efficient pre-processing, spatial segmentation, and classification.
License Artistic-2.0
Depends R (>= 4.3), ProtGenerics, BiocGenerics, BiocParallel, S4Vectors (>= 0.27.3), methods, stats, stats4
Imports CardinalIO, Biobase, EBImage, graphics, grDevices, irlba, Matrix, matter, nlme, parallel, utils
Suggests BiocStyle, testthat, knitr, rmarkdown
VignetteBuilder knitr
biocViews Software, Infrastructure, Proteomics, Lipidomics, MassSpectrometry, ImagingMassSpectrometry, ImmunoOncology, Normalization, Clustering, Classification, Regression
URL http://www.cardinalmsi.org
BugReports https://github.com/kuwisdelu/Cardinal/issues
git_url https://git.bioconductor.org/packages/Cardinal
git_branch RELEASE_3_19
git_last_commit a468515
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-01
Author Kylie Ariel Bemis [aut, cre]
Maintainer Kylie Ariel Bemis <k.bemis@northeastern.edu>
Contents

Cardinal-package .............................................. 3
bin .............................................................. 4
colocalized .................................................. 5
deprecated .................................................... 7
estimateDomain .............................................. 7
features ......................................................... 8
findNeighbors ................................................ 9
MassDataFrame-class ........................................ 10
MeansTest ...................................................... 11
MSImagingArrays-class ..................................... 13
MSImagingExperiment-class ............................... 15
normalize ....................................................... 17
peakAlign ...................................................... 18
peakPick ....................................................... 20
peakProcess ................................................... 22
pixels .......................................................... 24
plot-image ...................................................... 25
plot-spectra ................................................... 27
PositionDataFrame-class ..................................... 29
process ........................................................ 30
readMSIData .................................................... 32
recalibrate ..................................................... 34
reduceBaseline ............................................... 35
ResultsList-class ........................................... 37
selectROI ....................................................... 38
simulateSpectra ............................................. 39
slice ............................................................. 42
smooth ........................................................ 43
SpatialCV ....................................................... 45
SpatialDGMM .................................................. 46
SpatialFastmap ............................................... 49
SpatialKMeans ............................................... 51
SpatialNMF ..................................................... 53
SpatialPCA ..................................................... 54
SpatialPLS ..................................................... 56
SpatialResults-class ......................................... 58
SpatialShrunkenCentroids .................................. 60
spatialWeights ............................................... 63
SpectraArrays-class ......................................... 65
SpectrallImagingArrays-class ............................ 66
SpectrallImagingData-class ............................... 67
SpectrallImagingExperiment-class ....................... 69
spectrapply .................................................... 70
subsetFeatures .............................................. 72
summarizeFeatures ......................................... 73
writeMSIData .................................................. 75
Description

Implements statistical & computational tools for analyzing mass spectrometry imaging datasets, including methods for efficient pre-processing, spatial segmentation, and classification.

Details

Cardinal provides an abstracted interface to manipulating mass spectrometry imaging datasets, simplifying most of the basic programmatic tasks encountered during the statistical analysis of imaging data. These include image manipulation and processing of both images and mass spectra, and dynamic plotting of both.

While pre-processing steps including normalization, baseline correction, and peak-picking are provided, the core functionality of the package is statistical analysis. The package includes classification and clustering methods based on nearest shrunken centroids, as well as traditional tools like PCA and PLS.

Type `browseVignettes("Cardinal")` to view a user’s guide and vignettes of common workflows.

Options

The following options can be set:

- `getCardinalBPPARAM`, `setCardinalBPPARAM(BPPARAM=NULL)`: The default backend to use for parallel processing. By default, this is initially set to NULL (no parallelization). Otherwise, it must be a `BiocParallelParam` instance. See documentation for `bplapply`.
- `getCardinalVerbose()`, `setCardinalVerbose(VERBOSE=interactive())`: Should progress messages be printed?
- `getCardinalNChunks()`, `setCardinalNChunks(n=20L)`: The default number of data chunks used when iterating over large datasets. Used by many methods internally.

Additionally, graphical parameters can be set:

- `vizi_style()`: Set the default plotting style.
- `vizi_engine()`: Set the default plotting engine.
- `vizi_par()`: Set default graphical parameters.

Author(s)

Kylie A. Bemis
Maintainer: Kylie A. Bemis <k.bemis@northeastern.edu>
### Description

Apply on-the-fly binning to spectra.

### Usage

```r
## S4 method for signature 'MSImagingExperiment'
bin(x, ref,
    spectra = "intensity", index = "mz",
    method = c("sum", "mean", "max", "min",
               "linear", "cubic", "gaussian", "lanczos"),
    resolution = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'MSImagingArrays'
bin(x, ref,
    spectra = "intensity", index = "mz",
    method = c("sum", "mean", "max", "min",
               "linear", "cubic", "gaussian", "lanczos"),
    resolution = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'SpectralImagingExperiment'
bin(x, ref,
    spectra = "intensity", index = NULL,
    method = c("sum", "mean", "max", "min",
               "linear", "cubic", "gaussian", "lanczos"),
    resolution = NA, units = c("relative", "absolute"),
    verbose = getCardinalVerbose(), ...)

## S4 method for signature 'SpectralImagingArrays'
bin(x, ref,
    spectra = "intensity", index = NULL,
    method = c("sum", "mean", "max", "min",
               "linear", "cubic", "gaussian", "lanczos"),
    resolution = NA, units = c("relative", "absolute"),
    verbose = getCardinalVerbose(), ...)
```

### Arguments

- **x** A spectral imaging dataset.
- **ref** Optional. The bin centers, or their range if resolution is specified. Created from resolution if not provided.
- **spectra** The name of the array in `spectraData()` to bin.
colocalized

index
The name of the array in spectraData() (for SpectralImagingArrays) or column in featureData() (for SpectralImagingExperiment) to use for the bins.

method
The peak picking method to use. See approx1 for details.

resolution
The bin widths.

units
The units for the above resolution.

verbose
Should progress messages be printed?

... Ignored.

Details
The binning is applied but not processed immediately. It is performed on-the-fly whenever the spectra are accessed.

Value
A new object derived from SpectralImagingExperiment with the binned spectra.

Author(s)
Kylie A. Bemis

See Also
approx1, estimateDomain, estimateReferenceMz

Examples
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# bin to unit resolution
mse2 <- bin(mse, resolution=1, units="mz")

# bin to a specific range and resolution
mse3 <- bin(mse, ref=c(800, 1000), resolution=1, units="mz")

---

colocalized

Description
Find colocalized features in an imaging dataset.
Usage

```r
## S4 method for signature 'MSImagingExperiment'
colocalized(object, mz, ...)

## S4 method for signature 'SpectralImagingExperiment'
colocalized(object, i, ref, 
  threshold = median, n = Inf,
  sort.by = c("cor", "MOC", "M1", "M2", "Dice", "none"),
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpatialDGMM'
colocalized(object, ref, 
  threshold = median, n = Inf,
  sort.by = c("MOC", "M1", "M2", "Dice", "none"),
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM(), ...)
```

Arguments

- `object`: An imaging experiment.
- `mz`: An m/z value of a feature in `object` to use as a reference.
- `i`: The index of a feature in `object` to use as a reference.
- `ref`: Either a flattened image (i.e., a numeric vector) or a logical mask of a region-of-interest to use as a reference.
- `threshold`: Either a function that returns the cutoff to use for creating logical masks of numeric references, or a numeric threshold to use.
- `n`: The number of top-ranked colocalized features to return.
- `sort.by`: The colocalization measure used to rank colocalized features. Possible options include Pearson’s correlation (“cor”), Manders overlap coefficient (“MOC”), Manders colocalization coefficients (“M1” and “M2”), and Dice similarity coefficient (“Dice”).
- `nchunks`: The number of chunks to use for processing.
- `verbose`: Should progress messages be printed?
- `BPPARAM`: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- `...`: Options passed to `chunkApply`.

Value

A data frame with the colocalized features, or a list of data frames if multiple references are given.

Author(s)

Kylie A. Bemis
Examples

set.seed(1)
x <- simulateImage(preset=1, dim=c(10,10), representation="centroid")

# find features colocalized with first feature
colocalized(x, i=1)

deprecated

Deprecated and defunct objects in Cardinal

Description

These functions are provided for compatibility with older versions of Cardinal, and will be removed in the future.

estimateDomain

Estimate shared domain

Description

For unaligned spectral data, it is often necessary to estimate a suitable shared domain in order to calculate statistical summaries like the mean spectrum.

Usage

estimateDomain(xlist,
    units = c("relative", "absolute"),
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)

estimateReferenceMz(object,
    units = c("ppm", "mz"),
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)

estimateReferencePeaks(object, SNR = 2,
    method = c("diff", "sd", "mad", "quantile", "filter", "cwt"),
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)

Arguments

- **xlist**: A list of the domain values (e.g., m/z values) for each spectrum.
- **object**: A mass spectral imaging dataset.
- **units**: Should the spacing between domain values use absolute or relative units?
- **method**: The peak picking method to use. See `findpeaks` for details.
- **SNR**: The signal-to-noise threshold to use to determine a peak.
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- **...**: Options passed to `chunkLapply`.

Details

For `estimateDomain`, the domain is estimated by first finding the resolution of each spectrum’s individual domain values (e.g., the spacing between m/z values), and then creating a sequence of domain values using the median resolution of all spectra.

The `estimateReferenceMz` function simply calls `estimateDomain` on the appropriate components of a mass spectral imaging dataset to estimate profile m/z bins.

The `estimateReferencePeaks` function calculates the mean spectrum (or looks for a “mean” column in `featureData()`) and performs peak picking on the mean spectrum. It can be used to create a set of reference peaks if all relevant peaks appear in the mean spectrum.

Value

A vector of domain values, m/z values, or peaks.

Author(s)

Kylie A. Bemis

See Also

- `summarizeFeatures`, `recalibrate`, `peakAlign`, `peakProcess`

---

**features**

*Find feature indices*

Description

Search for the row indices of a spectral imaging dataset that correspond to specific or features, based on a set of conditions.
findNeighbors

Usage

## S4 method for signature 'MSImagingExperiment'
features(object, ..., mz, tolerance = NA, units = c("ppm", "mz"),
          env = NULL)

## S4 method for signature 'SpectralImagingExperiment'
features(object, ..., env = NULL)

Arguments

object A spectral imaging dataset.
... Expressions that evaluate to logical vectors in the environment of featureData().
mz The m/z values of features to include.
tolerance The tolerance for matching features to m/z values.
units The units for the above tolerance.
env The enclosing environment for evaluating ....

Author(s)

Kylie A. Bemis

Examples

set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))

features(mse, mz > 800, mz < 1800)
features(mse, mz=metadata(mse)$design$featureData$mz)

findNeighbors

Find spatial neighbors

Description

Find the indices of spatial neighbors for all observations in a dataset.

Usage

## S4 method for signature 'ANY'
findNeighbors(x, r = 1, groups = NULL,
              metric = "maximum", p = 2, matrix = FALSE, ...)

## S4 method for signature 'SpectralImagingData'
findNeighbors(x, r = 1, groups = run(x), ...)

## S4 method for signature 'PositionDataFrame'
findNeighbors(x, r = 1, groups = run(x), ...)
Arguments

- `x`: An imaging dataset or data frame with spatial dimensions.
- `r`: The spatial maximum distance for an observation to be considered a neighbor.
- `groups`: A vector coercible to a factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.
- `metric`: Distance metric to use when finding the nearest neighbors. Supported metrics include "euclidean", "maximum", "manhattan", and "minkowski".
- `p`: The power for the Minkowski distance.
- `matrix`: Should the neighbors be returned as a sparse adjacency matrix instead of a list?
- `...`: Additional arguments passed to the next call.

Value

Either a list of indices of neighbors or a sparse adjacency matrix (sparse_mat).

Author(s)

Kylie A. Bemis

See Also

spatialWeights

Examples

```r
pdata <- PositionDataFrame(coord=expand.grid(x=1:8, y=1:8))
# find spatial neighbors
findNeighbors(pdata, r=1)
```

MassDataFrame-class

MassDataFrame: Extended data frame with key columns

Description

A data frame for mass spectrometry feature metadata with a required column for m/z values.

Usage

```r
MassDataFrame(mz, ..., row.names = FALSE)
```
MeansTest

Arguments

mz  A sorted vector of m/z values.
...
row.names  Either a vector of row names or a logical value indicating whether row names
should be generated automatically (from the m/z values).

Methods

mz(object, ...), mz(object, ...) <- value: Get or set the m/z values.

Author(s)

Kylie A. Bemis

See Also

XDataFrame, PositionDataFrame

Examples

## Create an MassDataFrame object
MassDataFrame(mz=sort(500 * runif(10)), label=LETTERS[1:10])

MeansTest  Linear model-based testing for summarized imaging experiments

Description

Performs hypothesis testing for imaging experiments by fitting linear mixed models to summarizations or segmentations.

Usage

## S4 method for signature 'ANY'
meansTest(x, data, fixed, random, samples,
  response = "y", reduced = ~ 1, byrow = FALSE,
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
meansTest(x, fixed, random, samples = run(x),
  response = "intensity", ...)

## S4 method for signature 'SpatialDGMM'
meansTest(x, fixed, random, class = 1L,
  response = "intensity", reduced = ~ 1,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

segmentationTest(x, fixed, random, samples = run(x),
    class = 1L, response = "intensity", reduced = ~ 1, ...)

## S4 method for signature 'MeansTest'
topFeatures(object, n = Inf, sort.by = "statistic", ...)

## S4 method for signature 'MeansTest,missing'
plot(x, i = 1L, type = "boxplot", show.obs = TRUE,
    fill = FALSE, layout = NULL, ...)

Arguments

- **x**: A dataset in P x N matrix format or a set of spatially segmented images.
- **data**: A data frame of additional variables parallel to x.
- **fixed**: A one-sided formula giving the fixed effects of the model on the RHS. The response will added to the LHS, and the formula will be passed to the underlying modeling function.
- **random**: A one-sided formula giving the random effects of the model on the RHS. See \texttt{lme} for the allowed specifications.
- **samples**: A vector coercible to a factor giving the observational unit (i.e., the samples and replicates).
- **class**: For \texttt{SpatialDGMM}, the class (segment) from the Gaussian mixture models that should be used for the comparison. By default, compare the classes (segments) with the highest means in each sample.
- **response**: The name of to assign the response variable in the fitted models.
- **reduced**: A one-sided formula specifying the reduced model for the null hypothesis. The default is an intercept-only model.
- **byrow**: For the default method, are the rows or the columns the x?
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of \texttt{BiocParallelParam}. See documentation for \texttt{bplapply}.
- **...**: Passed to internal linear modeling methods. Either \texttt{lm} if only fixed effects are given or \texttt{lme} if random effects are given.
- **object**: A fitted model object to summarize.
- **n, sort.by**: For \texttt{topFeatures}, the number of top features to return and how to sort them.
- **i**: The index of the model(s)/feature(s) to plot.
- **type**: The type of plot to display.
- **show.obs**: Should individual observations (i.e., the summarized mean for each sample) be plotted too?
fill  Should the boxplots be filled?
layout  A vector of the form c(nrow, ncol) specifying the number of rows and columns in the facet grid.

Value

An object of class MeansTest derived from ResultsList, where each element contains a linear model.

Author(s)

Dan Guo and Kylie A. Bemis

See Also

lm, lme, spatialDGMM

Examples

set.seed(1)
x <- simulateImage(preset=4, nrun=3, npeaks=10,
dim=c(10,10), peakheight=5, peakdiff=2,
representation="centroid")
samples <- replace(run(x), !(x$circleA | x$circleB), NA)
fit <- meansTest(x, ~condition, samples=samples)
print(fit)
Arguments

spectraData Either a list-like object with lists of individual spectra and lists of their domain values, or a SpectraArrays instance.

pixelData A PositionDataFrame with pixel metadata, with a row for each spectrum.

experimentData Either NULL or a ImzMeta object with MS-specific experiment-level metadata.

centroided A logical value indicated whether the spectra have been centroided.

continuous A logical value indicated whether the spectra all have the same m/z values.

metadata A list of arbitrary metadata.

Slots

spectraData: A SpectraArrays object storing one or more array-like data elements with conformable dimensions.

elementMetadata: A PositionDataFrame containing spectrum-level metadata, including each spectrum’s pixel coordinates and experimental run information.

processing: A list containing unexecuted ProcessingStep objects.

experimentData: Either NULL or an ImzMeta object containing experiment-level metadata (necessary for writing the data to imzML).

centroided: A logical value indicated whether the spectra have been centroided (if known).

continuous: A logical value indicated whether the spectra all have the same m/z values (if known).

Methods

All methods for SpectralImagingData and SpectralImagingArrays also work on MSImagingArrays objects. Additional methods are documented below:

mz(object, i = NULL, ...), mz(object, i = NULL, ...) <- value: Get or set the m/z arrays in the spectraData slot.

intensity(object, i = NULL, ...), intensity(object, i = NULL, ...) <- value: Get or set the intensity arrays in the spectraData slot.

centroided(object, ...), centroided(object, ... <- value: Get or set the centroided slot.

isCentroided(object, ...): Equivalent to isTRUE(centroided(object)).

experimentData(object), experimentData(object) <- value: Get or set the experimentData slot.

Author(s)

Kylie A. Bemis

See Also

SpectralImagingArrays, MSImagingExperiment
**Examples**

```r
set.seed(1)
x <- replicate(9, rlnorm(10), simplify=FALSE)
mz <- replicate(9, 500 * sort(runif(10)), simplify=FALSE)
coord <- expand.grid(x=1:3, y=1:3)

msa <- MSImagingArrays(
    spectraData=list(intensity=x, mz=mz),
    pixelData=PositionDataFrame(coord))

print(msa)
```

**Description**

The MSImagingExperiment class provides a matrix-like container for high-throughput mass spectrometry imaging data where every mass spectrum shares the same m/z values. It is designed to provide easy access to both the spectra (as columns) and sliced images (as rows).

It can be converted from a MSImagingArrays object which is designed for representing raw mass spectra.

**Usage**

```r
## Instance creation
MSImagingExperiment(spectraData = SimpleList(),
    featureData = MassDataFrame(), pixelData = PositionDataFrame(),
    experimentData = NULL, centroided = NA, metadata = list())

## Additional methods documented below
```

**Arguments**

- `spectraData` Either a matrix-like object with number of rows equal to the number of features and number of columns equal to the number of pixels, a list of such objects, or a SpectraArrays instance.
- `featureData` A MassDataFrame with feature metadata, with a row for each feature.
- `pixelData` A PositionDataFrame with pixel metadata, with a row for each spectrum.
- `experimentData` Either NULL or a ImzMeta object with MS-specific experiment-level metadata.
- `centroided` A logical value indicated whether the spectra have been centroided.
- `metadata` A list of arbitrary metadata.
Slots

spectraData: A `SpectraArrays` object storing one or more array-like data elements with conformable dimensions.

featureData: A `MassDataFrame` containing feature-level metadata.

elementMetadata: A `PositionDataFrame` containing spectrum-level metadata, including each spectrum’s pixel coordinates and experimental run information.

processing: A list containing unexecuted `ProcessingStep` objects.

experimentData: Either NULL or an `ImzMeta` object containing experiment-level metadata (necessary for writing the data to imzML).

centroided: A logical value indicated whether the spectra have been centroided (if known).

Methods

All methods for `SpectralImagingData` and `SpectralImagingExperiment` also work on `MSImagingExperiment` objects. Additional methods are documented below:

`mz(object, ...), mz(object, ...) <- value`: Get or set the m/z column in the `featureData` slot.

`intensity(object, ...), intensity(object, ...) <- value`: Get or set the intensity matrix in the `spectraData` slot.

`centroided(object, ...), centroided(object, ...) <- value`: Get or set the `centroided` slot.

`isCentroided(object, ...): Equivalent to isTRUE(centroided(object))`.

`experimentData(object), experimentData(object) <- value`: Get or set the `experimentData` slot.

Author(s)

Kylie A. Bemis

See Also

`SpectralImagingExperiment, MSImagingArrays`

Examples

```r
set.seed(1)
x <- matrix(rlnorm(81), nrow=9, ncol=9)
mz <- sort(runif(9))
coord <- expand.grid(x=1:3, y=1:3)

mse <- MSImagingExperiment(
spectraData=x,
featureData=MassDataFrame(mz=mz),
pixelData=PositionDataFrame(coord))

print(mse)
```
**normalize**

**Normalize spectra**

**Description**

Apply deferred normalization to spectra.

**Usage**

```r
## S4 method for signature 'SpectralImagingData'
normalize(object,
          method = c("tic", "rms", "reference"), ...)
```

**Arguments**

- **object**: A spectral imaging dataset.
- **method**: The normalization method to use. See `rescale` for details.
- **...**: Additional arguments passed to the normalization function.

**Details**

The supported normalization methods are:

- "tic": Total ion current normalization using `rescale_sum`.
- "rms": Root-mean-squared normalization using `rescale_rms`.
- "reference": Normalization according to a reference feature using `rescale_ref`.

**Value**

An object of the same class with the processing step queued.

**Note**

The normalization is deferred until `process()` is called.

**Author(s)**

Kylie A. Bemis

**See Also**

`smooth`, `recalibrate`, `reduceBaseline`, `peakPick`, `process`
 Examples

set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# queue normalization
mse2 <- normalize(mse, method="tic")

# apply normalization
mse2 <- process(mse2)

 peakAlign

 Align peaks across spectra

 Description

Align peaks across spectra in a spectral imaging dataset.

 Usage

## S4 method for signature 'MSImagingExperiment'
peakAlign(object, ref,
spectra = "intensity", index = "mz",
tolerance = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'MSImagingArrays'
peakAlign(object, ref,
spectra = "intensity", index = "mz",
tolerance = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'SpectralImagingExperiment'
peakAlign(object, ref,
spectra = "intensity", index = NULL,
tolerance = NA, units = c("relative", "absolute"),
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingArrays'
peakAlign(object, ref,
spectra = "intensity", index = NULL,
tolerance = NA, units = c("relative", "absolute"),
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)
peakAlign

Arguments

- **object**: A spectral imaging dataset.
- **ref**: The locations of reference peaks to use for the alignment.
- **spectra**: The name of the array in `spectraData()` to use for the peak intensities.
- **index**: The name of the array in `spectraData()` (for `SpectralImagingArrays`) or column in `featureData()` (for `SpectralImagingExperiment`) to use for the peak locations.
- **tolerance**: The tolerance for matching a detected peak to the reference. If `NA`, then the tolerance is automatically determined as half the minimum distance between locations in the estimated spectral domain (see "Details").
- **units**: The units for the above tolerance.
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- **...**: Options passed to `process()`.

Details

Before peak alignment, `process()` is called to apply any queued pre-processing steps. It is assumed that `peakPick()` has either been queued or already applied to the data.

If **ref** is provided, then the aligned peaks are returned immediately without additional processing. (Peaks are binned on-the-fly to the reference peak locations.)

If **ref** is not provided, then the shared peaks must be determined automatically. This starts with creation of a shared domain giving a list of possible peak locations. For `SpectralImagingArrays`, `estimateDomain()` is used to create the shared domain from the index array. For `SpectralImagingExperiment`, the index column of `featureData()` is used directly.

Next, **binpeaks** is used to bin the observed peaks to the shared domain. Then, **mergepeaks** is used to merge peaks that are separated by a distance less than the given tolerance.

The averaged locations of the merged peaks in each bin are used as the shared peaks for the full dataset, and the aligned peaks are returned. (Peaks are binned on-the-fly to the shared peak locations.)

Value

A new object derived from `SpectralImagingExperiment` with the aligned peaks.

Author(s)

Kylie A. Bemis

See Also

`process, peakPick, peakProcess`
Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# queue peak picking
mse2 <- peakPick(mse, method="diff", SNR=6)

# align peaks
mse2 <- peakAlign(mse2)
plot(mse2, i=4)
```

---

### peakPick

**Peak pick spectra**

#### Description

Apply deferred peak picking to spectra.

#### Usage

```r
## S4 method for signature 'MSImagingExperiment'
peakPick(object, ref,
  method = c("diff", "sd", "mad", "quantile", "filter", "cwt"),
  SNR = 2, type = c("height", "area"),
  tolerance = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'MSImagingArrays'
peakPick(object, ref,
  method = c("diff", "sd", "mad", "quantile", "filter", "cwt"),
  SNR = 2, type = c("height", "area"),
  tolerance = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'SpectralImagingData'
peakPick(object, ref,
  method = c("diff", "sd", "mad", "quantile", "filter", "cwt"),
  SNR = 2, type = c("height", "area"),
  tolerance = NA, units = c("relative", "absolute"), ...)
```

#### Arguments

- **object**: A spectral imaging dataset.
- **ref**: Optional vector giving locations of reference peaks to extract from the dataset.
- **method**: The peak picking method to use. See `findpeaks` for details.
- **SNR**: The signal-to-noise threshold to use to determine a peak.
- **type**: The type of value to use to summarize the peak.
peakPick

tolerance  If `ref` is specified, the tolerance to use when deciding if a local peak in a spectrum matches a reference peak. If `NA`, then the tolerance is automatically determined as half the minimum distance between peaks in the reference.
units  The units for the above tolerance.
...  Additional arguments passed to the peak picking function.

Details

Unless otherwise specified, peaks are detected as local maxima which are then compared to the estimated noise level to determine a signal-to-noise ratio for each peak. Most of the peak detection methods below are differentiated by how they estimate the noise in the spectrum.

The supported peak picking methods are:

- "diff": Estimate noise based on the derivative of the signal using `estnoise_diff`.
- "sd": Estimate noise from standard deviation using `estnoise_sd`.
- "mad": Estimate noise from mean absolute deviation using `estnoise_mad`.
- "quantile": Estimate noise from a rolling quantile of the difference between the raw signal and a smoothed signal using `estnoise_quant`.
- "filter": Estimate noise using dynamic filtering of the local peaks using `estnoise_filt`.
- "cwt": Detect peaks based on continuous wavelet transform (CWT) using `findpeaks_cwt`.

If `ref` is provided, then the signal-to-noise ratio is not determined, and any detected local maxima are summarized as long as they match to a reference peak.

Value

An object of the same class with the processing step queued.

Note

The peak picking is deferred until `process()` is called.

Author(s)

Kylie A. Bemis

See Also

`process`, `peakAlign`, `peakProcess`, `estimateReferencePeaks`

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# queue peak picking
mse2 <- peakPick(mse, method="diff", SNR=6)
plot(mse2, i=4)
```
# apply peak picking
mse2 <- process(mse2)

## peakProcess
Process peaks in mass spectra

### Description
Apply peak picking and alignment to a mass spectrometry imaging dataset.

### Usage
```r
## S4 method for signature 'MSImagingExperiment_OR_Arrays'
peakProcess(object, ref,
    spectra = "intensity", index = "mz",
    method = c("diff", "sd", "mad", "quantile", "filter", "cwt"),
    SNR = 2, type = c("height", "area"),
    tolerance = NA, units = c("ppm", "mz"),
    sampleSize = NA, filterFreq = TRUE, outfile = NULL,
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)
```

### Arguments
- **object**: A spectral imaging dataset.
- **ref**: The locations of reference peaks to use for the alignment.
- **spectra**: The name of the array in `spectraData()` to use for the peak intensities.
- **index**: The name of the array in `spectraData()` (for `MSImagingArrays`) or column in `featureData()` (for `MSImagingExperiment`) to use for the peak locations.
- **method**: The peak picking method to use. See `findpeaks` for details.
- **SNR**: The signal-to-noise threshold to use to determine a peak.
- **type**: The type of value to use to summarize the peak.
- **tolerance**: The tolerance for matching a detected peak to the reference peaks or the shared m/z values. Passed to `peakPick` and `peakAlign`.
- **units**: The units for the above tolerance.
- **sampleSize**: The count or proportion giving a subset of spectra to use to determine reference peaks.
- **filterFreq**: Either a logical value indicating whether singleton peaks should be removed, or a count or frequency used as a threshold to filter the peaks.
- **outfile**: Optional. The name of a file to write the resulting dataset as imzML.
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- **...**: Options passed to `process()`.
Details

This method provides a combined interface for `peakPick` and `peakAlign` for the most common approaches to peak processing.

If `peakPick()` has been queued already, then it will be applied. Otherwise, it will be called internally with the provided arguments.

There are two main paths depending on whether (1) peaks should be extracted based on a reference or (2) peak picking should be performed on the full dataset and then aligned.

If either `ref` is provided or `sampleSize` is finite, then (1) is chosen and peaks are extracted based on the reference. If the reference is not provided, then peak picking and alignment performed on a subset of spectra (according to `sampleSize`) to create the reference peaks. The reference peaks are then used to extract peaks from the full dataset.

Otherwise, (2) is chosen and peaks are picked and aligned across all spectra.

The advantage of (1) is that all reference peaks will be summarized even they would not have a high enough signal-to-noise ratio to be detected in some spectra. The disadvantage is that rare peaks that do not appear in the sampled subset of spectra will not be included in the process peaks.

The advantage of (2) is that rare peaks will be included because peak detection is performed on all spectra. The disadvantage is that some peaks may be missing from some spectra despite having nonzero intensities, because they did not have a high enough signal-to-noise ratio to be detected as peaks.

Setting `sampleSize` to 1 will balance these advantages and disadvantages because the reference will be based on all spectra. However, this means the full dataset must be processed at least twice (possibly more if intermediate calculations are necessary), so it will be more time-consuming.

Value

A new object derived from `MSImagingExperiment` with the processed peaks.

Author(s)

Kylie A. Bemis

See Also

`process peakPick, peakAlign`

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# process peaks
mse2 <- peakProcess(mse, method="diff", SNR=3)
plot(mse2, i=4)
```
Find pixel indices

Description

Search for the column indices of a spectral imaging dataset that correspond to specific pixels, based on a set of conditions.

Usage

```r
## S4 method for signature 'SpectralImagingExperiment'
pixels(object, ..., coord, run, tolerance = NA,
       env = NULL)

## S4 method for signature 'SpectralImagingArrays'
pixels(object, ..., coord, run, tolerance = NA,
       env = NULL)

## S4 method for signature 'SpectralImagingData'
pixels(object, ..., env = NULL)
```

Arguments

- `object` A spectral imaging dataset.
- `...` Expressions that evaluate to logical vectors in the environment of `pixelData()`.
- `coord` The coordinates of the pixels to include.
- `run` The run of the pixels to include.
- `tolerance` The tolerance for matching pixels to coordinates.
- `env` The enclosing environment for evaluating `...`.

Author(s)

Kylie A. Bemis

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))
pixels(mse, x > 6, y > 6)
pixels(mse, coord=expand.grid(x=1:3, y=1:3))
```
Description

Create and display images from the spectra or pixel data of a spectral imaging dataset using a formula interface.

Usage

```r
## S4 method for signature 'MSImagingExperiment'
image(x,
      formula = intensity ~ x * y,
      i = features(x, mz=mz),
      mz = NULL,
      tolerance = NA,
      units = c("ppm", "mz"),
      ...,
      xlab, ylab)
```

```r
## S4 method for signature 'SpectralImagingExperiment'
image(x,
      formula,
      i = 1L,
      run = NULL,
      groups = NULL,
      superpose = FALSE,
      key = TRUE,
      ...,
      enhance = NULL,
      smooth = NULL,
      scale = NULL,
      subset = TRUE)
```

Arguments

- `x`: A spectral imaging dataset.
- `formula`: A formula of the form `vals ~ x * y` giving the image values and the pixel coordinates. The LHS is taken to be the name of an array in `spectraData()` and the RHS is taken to be columns of `pixelData()`. Alternatively, if `formula` is a string or if `i` is `NULL`, then the LHS is interpreted as the name of a column of `pixelData()` as well.
- `i`: The index of the feature(s) to plot for the image(s).
- `mz`: The m/z value(s) to plot for the image(s).
- `tolerance`: If specified, the tolerance to consider a feature as being equal to the given m/z value.
units

The units for the above tolerance.

Additional arguments passed to `plot_image`.

xlab, ylab

Plotting labels.

run

The names of experimental runs to include, or the index of the levels of the runs to include.

groups

A vector coercible to a factor indicating which of the specified features should be plotted with the same color.

superpose

If multiple images are plotted, should they be superposed on top of each other, or plotted separately?

key

Should a legend or colorkey be plotted?

enhance

The name of a contrast enhancement method, such as "hist" or "adapt" for `enhance_hist()` and `enhance_adapt()`, etc. See `enhance` for details.

smooth

The name of a smoothing method, such as "gauss" or "bi" for `filt2_gauss()` and `filt2_bi()`, etc. See `filt2` for details.

scale

If multiple images are plotted, should they be scaled to the same intensity scale?

subset

A logical vector indicating which pixels to include in the image.

Author(s)

Kylie A. Bemis

See Also

`image, plot_image, selectROI`

Examples

```r
set.seed(1)
x <- simulateImage(preset=2, npeaks=10, dim=c(16,16))
peaks <- mz(metadata(x)$design$featureData)

image(x, mz=peaks[1L], tolerance=0.5, units="mz")
image(x, mz=peaks[1L], smooth="gaussian")
image(x, mz=peaks[1:9], smooth="adaptive")

x <- summarizePixels(x, stat=c(TIC="mean"))
image(x, "TIC")
```
Description

Create and display sliced images from a spectral imaging dataset using a formula interface.

Usage

```r
## S4 method for signature 'MSImagingExperiment,missing'
plot(x,
     formula = intensity ~ mz,
     i = pixels(x, coord=coord, run=run),
     coord = NULL,
     run = NULL,
     ...
     xlab, ylab,
     isPeaks = isCentroided(x))

## S4 method for signature 'MSImagingArrays,missing'
plot(x,
     formula = intensity ~ mz,
     i = pixels(x, coord=coord, run=run),
     coord = NULL,
     run = NULL,
     ...
     xlab, ylab,
     isPeaks = isCentroided(x))

## S4 method for signature 'SpectralImagingExperiment,missing'
plot(x,
     formula,
     i = 1L,
     groups = NULL,
     superpose = FALSE,
     key = TRUE,
     ...
     n = Inf,
     downsampler = "lttb",
     isPeaks = FALSE,
     annPeaks = 0)

## S4 method for signature 'SpectralImagingArrays,missing'
plot(x,
     formula,
     i = 1L,
     groups = NULL,
superpose = FALSE,
key = TRUE,
..., 
n = Inf,
downsampler = "lttb",
isPeaks = FALSE,
annPeaks = 0)

Arguments

- **x**: A spectral imaging dataset.
- **formula**: A formula of the form `vals ~ t` giving the spectra values and their domain locations. The LHS is taken to be the name of an array in `spectraData()` and the RHS is either an array in `spectraData()` for `SpectralImagingArrays`-derived classes or a column of `featureData()` for `SpectralImagingExperiment`-derived classes. Alternatively, if `formula` is a string or if `i` is NULL, then the LHS is interpreted as the name of a column of `featureData()` for `SpectralImagingExperiment` as well.
- **i**: The index of the spectrum to plot.
- **coord**: The coordinates of the spectrum to plot.
- **run**: The run of the spectrum to plot.
- **...**: Additional arguments passed to `plot_signal`.
- **xlab, ylab**: Plotting labels.
- **isPeaks**: Should the spectrum be plotted as peaks or as a continuous signal?
- **annPeaks**: If `isPeaks` is TRUE, either an integer giving the number of peaks to annotate (i.e., label with their location), or a plotting symbol (e.g., "circle", "cross", etc.) to indicate the peak locations.
- **groups**: A vector coercible to a factor indicating which of the specified spectra should be plotted with the same color.
- **superpose**: If multiple spectra are plotted, should they be superposed on top of each other, or plotted separately?
- **key**: Should a legend or color key be plotted?
- **n, downsampler**: A spectrum can contain far more data points than are needed to visualize it, potentially making the plotting unnecessarily slow. Downsampling can be performed to improve plotting speed while maintaining the visual integrity of the spectrum. See `downsample` for details.

Author(s)

Kylie A. Bemis

See Also

`plot`, `plot_signal`
Examples

```r
set.seed(1)
x <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

plot(x, i=4)
plot(x, coord=c(x=1, y=2))
plot(x, log2(intensity + 1) ~ mz, i=4,
     xlab=expression(italic("m/z")),
     ylab=expression(italic("Log Intensity")))
```

PositionDataFrame-class

PositionDataFrame: Extended data frame with key columns

Description

A data frame for metadata with spatial coordinates and multiple experimental runs.

Usage

```
PositionDataFrame(coord, run, ..., row.names = FALSE)
```

Arguments

- **coord**: A data frame or matrix of coordinates.
- **run**: A factor giving the experimental runs.
- **...**: Arguments passed to the `DataFrame()`.
- **row.names**: Either a vector of row names or a logical value indicating whether row names should be generated automatically (from the m/z values).

Methods

- `coord(object), coord(object) <- value`: Get or set the coordinate columns.
- `coordNames(object), coordNames(object) <- value`: Get or set the names of the coordinate columns.
- `run(object), run(object) <- value`: Get or set the experimental run column.
- `runNames(object), runNames(object) <- value`: Get or set the experimental run levels.
- `nrun(object)`: Get the number of experimental runs.
- `is3D(object)`: Check if the number of spatial dimensions is greater than 2.

Author(s)

Kylie A. Bemis
process

See Also

XDataFrame, MassDataFrame

Examples

## Create an PositionDataFrame object
coord <- expand.grid(x=1:3, y=1:3)
PositionDataFrame(coord=coord, label=LETTERS[1:9])

---

**process**

*Apply queued processing to spectra*

Description

Queue pre-processing steps on an imaging dataset and apply them, possibly writing out the processed data to a file.

Usage

```r
## S4 method for signature 'MSImagingExperiment'
process(object, spectra = "intensity", index = "mz",
  domain = NULL, outfile = NULL, ...)

## S4 method for signature 'MSImagingArrays'
process(object, spectra = "intensity", index = "mz",
  domain = NULL, outfile = NULL, ...)

## S4 method for signature 'SpectralImagingExperiment'
process(object, spectra = "intensity", index = NULL,
  domain = NULL, outfile = NULL,
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingArrays'
process(object, spectra = "intensity", index = NULL,
  domain = NULL, outfile = NULL,
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingData'
addProcessing(object, FUN, label, verbose = getCardinalVerbose(), ...)

reset(object, ...)
```
### Arguments

- **object**: A spectral imaging dataset.
- **spectra**: The name of the array in `spectraData()` to use for the peak intensities.
- **index**: The name of the array in `spectraData()` (for `MSImagingArrays`) or column in `featureData()` (for `MSImagingExperiment`) to use for the peak locations.
- **domain**: Optional. The name of the array in `spectraData()` (for `MSImagingArrays`) or column in `featureData()` (for `MSImagingExperiment`) to use for output domain (if known).
- **outfile**: Optional. The name of a file to write the resulting dataset. Creates an imzML file for `MSImagingExperiment` or `MSImagingArrays`. The "continuous" format will be written if domain is specified; otherwise the "processed" format will be used in most cases.
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- **...**: Options passed to `chunk_mapply` or `chunk_colapply`.
- **FUN**: A user-specified processing function.
- **label**: The name of the processing step.

### Details

This method allows queue of delayed processing to an imaging dataset. All of the queued processing steps will be applied in sequence whenever `process()` is called next. Use `reset()` to remove all queued processing steps.

Typically, processing steps are queued using methods like `normalize`, `smooth`, `peakPick`, etc. However, a processing step can be queued manually with `addProcessing`.

In this case, the user-specified function must accept (1) a first argument giving the spectral intensities as a numeric vector and (2) a second argument giving the intensity locations (e.g., m/z values) as a numeric vector.

The value returned by a user-specified function must return either (1) a numeric vector of the same length as the input intensities or (2) a 2-column matrix where the first column is the new locations (e.g., m/z values of peaks) and the second column is the new intensities.

### Value

An object of the same class as the original object, with all processing steps applied.

### Author(s)

Kylie A. Bemis

### See Also

`normalize`, `smooth`, `recalibrate`, `reduceBaseline`, `peakPick`
Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, dim=c(3,3), baseline=1)

mse2 <- smooth(mse, width=11)
mse2 <- reduceBaseline(mse2)
plot(mse2, i=4)

mse2 <- process(mse2)
```

---

**readMSIData**

Read mass spectrometry imaging data files

**Description**

Read supported mass spectrometry imaging data files, including imzML and Analyze 7.5.

**Usage**

```r
## Read any supported MS imaging file
readMSIData(file, ...)

## Read imzML file
readImzML(file, memory = FALSE, check = FALSE,
mass.range = NULL, resolution = NA, units = c("ppm", "mz"),
guess.max = 1000L, as = "auto", parse.only=FALSE,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## Read Analyze 7.5 file
readAnalyze(file, memory = FALSE, as = "auto",
verbose = getCardinalVerbose(), ...)

## Convert from MSImagingExperiment to MSImagingArrays
convertMSImagingExperiment2Arrays(object)

## Convert from MSImagingArrays to MSImagingExperiment
convertMSImagingArrays2Experiment(object, mz = NULL,
mass.range = NULL, resolution = NA, units = c("ppm", "mz"),
guess.max = 1000L, tolerance = 0.5 * resolution,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)
```
Arguments

- **file**: The absolute or relative file path. The file extension must be included for `readMSIData`.
- **memory**: Should the spectra be loaded into memory? If FALSE, the spectra are attached as an out-of-memory matrix.
- **check**: Should the UUID and checksum of the binary data file be checked against the corresponding imzML tags?
- **mass.range**: The mass range to use when converting the data to an `MSImagingExperiment`.
- **resolution**: The mass resolution to use when converting the data to an `MSImagingExperiment`. This is the inverse of the instrument resolution, if known. It is the width of the m/z bins when converting the data to an `MSImagingExperiment`.
- **units**: The units for the above resolution.
- **guess.max**: The number of spectra to use when guessing the mass range and resolution, if they are not provided.
- **as**: After reading in the data, what class of object should be returned? The data is initially loaded as an `MSImagingArrays` object. It may be converted to an `MSImagingExperiment` object. Setting to “auto” means to determine whichever is more appropriate depending on whether the spectra appear to have been processed and centroided.
- **parse.only**: If TRUE, return only the parsed imzML metadata without creating a new `MSImagingArrays` or `MSImagingExperiment` object.
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of `BiocParallelParam`. See documentation for `bplapply`.
- **...**: Additional arguments passed to `parseImzML` or `parseAnalyze`.
- **object**: A mass spectrometry imaging dataset to convert from one class to another.
- **mz**: A vector of shared m/z values for converting to `MSImagingExperiment`, if not to be determined automatically.
- **tolerance**: If the spectra have been centroided but the peaks are unaligned, then this is passed to `peakAlign`.

Details

The spectra are initially loaded into a `MSImagingArrays` object before conversion to `MSImagingExperiment` (if applicable).

This conversion can be sped up by specifying the `mass.range` and `resolution` so they do not have to be determined from the spectra directly. Using a larger value of `guess.max` can improve the accuracy of the m/z binning for downstream analysis at the expense of a longer conversion time.

If greater control is desired, spectra should be imported as `MSImagingArrays`, and processing to `MSImagingExperiment` can be performed manually.

If problems are encountered while trying to import imzML files, the files should be verified and fixed with imzMLValidator.

A Java version of imzML validator can be found at: [https://gitlab.com/imzML/imzMLValidator](https://gitlab.com/imzML/imzMLValidator).

A web-based version of imzML validator can be found at: [https://imzml.github.io](https://imzml.github.io).
Recalibrate spectra

Description

Apply deferred recalibration to spectra.

Usage

```r
## S4 method for signature 'MSImagingExperiment_OR_Arrays'
recalibrate(object, ref,
  method = c("locmax", "dtw", "cow"),
  tolerance = NA, units = c("ppm", "mz"), ...)

## S4 method for signature 'SpectralImagingData'
recalibrate(object, ref,
  method = c("locmax", "dtw", "cow"),
  tolerance = NA, units = c("relative", "absolute"), ...)
```

Arguments

- `object`: A spectral imaging dataset.
- `ref`: The domain (m/z) values or indices of reference peaks to use for the recalibration.
- `method`: The recalibration method to use. See `warp1` for details.
- `tolerance`: The tolerance for how much a peak can be shifted in either direction.
- `units`: The units for the above tolerance.
- `...`: Additional arguments passed to the recalibration function.
reduceBaseline

Details
The supported recalibration methods are:

- "locmax": Align to local maxima using warp1_loc.
- "dtw": Dynamic time warping using warp1_dtw.
- "cow": Correlation optimized warping using warp1_cow.

Value
An object of the same class with the processing step queued.

Note
The recalibration is deferred until process() is called.

Author(s)
Kylie A. Bemis

See Also
normalize, smooth, recalibrate, peakPick, process

Examples
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3), sdmz=250)
plot(mse, i=c(2,4,5), superpose=TRUE, xlim=c(1260,1320))

# queue recalibration
peaks <- estimateReferencePeaks(mse)
mse2 <- recalibrate(mse, ref=peaks, method="locmax", tolerance=500)

# apply recalibration
mse2 <- process(mse2)
plot(mse2, i=c(2,4,5), superpose=TRUE, xlim=c(1260,1320))
Arguments

- **object**: A spectral imaging dataset.
- **method**: The baseline estimation method to use. See `estbase` for details.
- **...**: Additional arguments passed to the baseline estimation function.

Details

The supported baseline estimation methods are:

- "locmin": Interpolate from local minima using `estbase_loc`.
- "hull": Convex hull estimation using `estbase_hull`.
- "snip": Sensitive nonlinear iterative peak (SNIP) clipping using `estbase_snip`.
- "median": Running medians using `estbase_med`.

Value

An object of the same class with the processing step queued.

Note

The baseline reduction is deferred until `process()` is called.

Author(s)

Kylie A. Bemis

See Also

`normalize`, `smooth`, `reduceBaseline`, `peakPick`, `process`

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3), baseline=1)

# queue baseline reduction
mse2 <- reduceBaseline(mse, method="locmin")
plot(mse2, i=4)

# apply baseline reduction
mse2 <- process(mse2)
```
Description

The `ResultsList` class provides a container for modeling results with spatial metadata.

Usage

```r
## Instance creation
ResultsList(..., mcols = NULL)
```

## Additional methods documented below

Arguments

- `...`: The modeling results.
- `mcols`: The metadata columns.

Methods

All methods for `SimpleList` also work on `ResultsList` objects. Additional methods are documented below:

- `fitted(object, ...)`: Extract fitted values from each modeling results object in the list.
- `predict(object, ...)`: Predict on each modeling results object in the list.
- `topFeatures(object, ...)`: Rank top features for each modeling results object in the list.
- `plot(x, i = 1L, ...)`: Plot the i-th modeling results.
- `image(x, i = 1L, ...)`: Display images for the i-th modeling results.

Author(s)

Kylie A. Bemis

See Also

`SpatialResults`
selectROI

Select regions-of-interest in an image

Description

Manually select regions-of-interest or pixels on an imaging dataset. The selectROI method uses the built-in locator function. It can be used with an existing image plot, or a new image will be plotted if image arguments are passed via `...`.

The regions of interest are returned as logical vectors indicating which pixels have been selected. These logical vectors can be combined into factors using the makeFactor function.

Usage

```r
## S4 method for signature 'SpectralImagingExperiment'
selectROI(object, ..., mode = c("region", "pixels"))

makeFactor(..., ordered = FALSE)
```

Arguments

- `object` A spectral imaging dataset.
- `mode` The mode of selection: "region" to select a region-of-interest as a polygon, or "pixels" to select individual pixels.
- `...` Additional arguments to be passed to `image` for `selectROI`, or name-value pairs of logical vectors to be combined by `makeFactor`.
- `ordered` Should the resulting factor be ordered or not?

Value

A logical vector of length equal to the number of pixels for `selectROI`.

A factor of the same length as the logical vectors for `makeFactor`.

Author(s)

Kylie A. Bemis

See Also

- `image`
**simulateSpectra**

_Simulate a mass spectrum or MS imaging experiment_

**Description**

Simulate mass spectra or complete MS imaging experiments, including a possible baseline, spatial and spectral noise, mass drift, mass resolution, and multiplicative variation, etc.

A number of preset imaging designs are available for quick-and-dirty simulation of images.

These functions are designed for small proof-of-concept examples and testing, and may not scale well to simulating larger datasets.

**Usage**

```r
simulateSpectra(n = 1L, npeaks = 50L, 
mz = rlnorm(npeaks, 7, 0.3), intensity = rlnorm(npeaks, 1, 0.9), 
from = 0.9 * min(mz), to = 1.1 * max(mz), by = 400, 
sdpeaks = sdpeakmult * log1p(intensity), sdpeakmult = 0.2, 
sdnoise = 0.1, sdmz = 10, resolution = 1000, fmax = 0.5, 
baseline = 0, decay = 10, units=c("ppm", "mz"), 
representation = c("profile", "centroid"), ...)
```

```r
simulateImage(pixelData, featureData, preset, 
from = 0.9 * min(mz), to = 1.1 * max(mz), by = 400, 
sdrun = 1, sdpixel = 1, spcorr = 0.3, units=c("ppm", "mz"), 
representation = c("profile", "centroid"), 
nchunks = getCardinalNChunks(), 
verbose = getCardinalVerbose(), 
BPPARAM = getCardinalBPPARAM(), ...)
```

```r
addShape(pixelData, center, size, shape=c("circle", "square"), name=shape)
```

```r
presetImageDef(preset = 1L, nrun = 1, npeaks = 30L, 
dim = c(20L, 20L), peakheight = 1, peakdiff = 1, 
sdsample = 0.2, jitter = TRUE, ...)
```

**Arguments**

- `n` The number of spectra to simulate.
- `npeaks` The number of peaks to simulate. Not used if `mz` and `intensity` are provided.
- `mz` The theoretical m/z values of the simulated peaks.
- `intensity` The mean intensities of the simulated peaks.
- `from` The minimum m/z value used for the mass range.
- `to` The maximum m/z value used for the mass range.
- `by` The step-size used for the observed m/z-values of the profile spectrum.
sdpeaks: The standard deviation(s) for the distributions of observed peak intensities on the log scale.

sdpeakmult: A multiplier used to calculate sdpeaks based on the mean intensities of peaks; used to simulate multiplicative variance. Not used if sdpeaks is provided.

sdnoise: The standard deviation of the random noise in the spectrum on the log scale.

sdmz: The standard deviation of the mass error in the observed m/z values of peaks, in units indicated by units.

resolution: The mass resolution as defined by \( \frac{m}{dm} \), where \( m \) is the observed mass and \( dm \) is the width of the peak at a proportion of its maximum height defined by fmax (defaults to full-width-at-half-maximum – FWHM – definition). Note that this is NOT the same as the definition of resolution in the readImzML function.

fmax: The fraction of the maximum peak height to use when defining the mass resolution.

baseline: The maximum intensity of the baseline. Note that baseline=0 means there is no baseline.

decay: A constant used to calculate the exponential decay of the baseline. Larger values mean the baseline decays more sharply at the lower mass range of the spectrum.

units: The units for by and sdmz. Either parts-per-million or absolute m/z units.

representation: Should a profile spectrum be returned or only the centroided peaks?

nchunks: The number of chunks to use for processing.

verbose: Should progress messages be printed?

BPPARAM: An optional instance of BiocParallelParam. See documentation for bplapply.

pixelData: A PositionDataframe giving the pixel design of the experiment. The names of the columns should match the names of columns in featureData. Each column should be a logical vector corresponding to a morphological substructure, indicate which pixels belong to that substructure.

featureData: A MassDataFrame giving the feature design of the experiment. Each row should correspond to an expected peak. The names of the columns should match the names of columns in pixelData. Each column should be a numeric vector corresponding to a morphological substructure, giving the mean intensity of that peak for that substructure.

preset: A number indicating a preset image definition to use.

nrun: The number of runs to simulate for each condition.

sdrun: A standard deviation giving the run-to-run variance.

sdpixel: A standard deviation giving the pixel-to-pixel variance.

spcorr: The spatial autocorrelation. Must be between 0 and 1, where spcorr=0 indicates no spatial autocorrelation.

...: Additional arguments to pass to simulateSpectra or presetImageDef.

dim: The dimensions of the preset image.

peakheight: Reference intensities used for peak heights by the preset.

peakdiff: A reference intensity difference used for the mean peak height difference between conditions, for presets that simulate multiple conditions.
simulateSpectra

sdsample  A standard deviation giving the amount of variation from the true peak heights for this simulated sample.

jitter    Should random noise be added to the location and size of the shapes?

center    The center of the shape.

size      The size of the shape (from the center).

shape     What type of shape to add.

name      The name of the added column.

Details

The simulateSpectra() and simulateImage() functions are used to simulate mass spectra and MS imaging experiments. They provide a great deal of control over the parameters of the simulation, including all sources of variation.

For simulateImage(), the user should provide the design of the simulated experiment via matching columns in pixelData and featureData, where each column corresponds to different morphological substructures or differing conditions. These design data frames are returned in the metadata() of the returned object for later reference.

A number of presets are defined by presetImageDef(), which returns only the pixelData and featureData necessary to define the experiment for simulateImage(). These can be referenced for help in understanding how to define experiments for simulateImage().

The preset images are:

• 1: a centered circle
• 2: a topleft circle and a bottomright square
• 3: two corner squares and a centered circle
• 4: a centered circle with conditions A and B in different runs
• 5: a topleft circle and a bottomright square with conditions A and B in different runs
• 6: two corner squares and a centered circle; the circle has conditions A and B in different runs
• 7: matched pairs of circles with conditions A and B within the same runs; includes reference peaks
• 8: matched pairs of circles inside squares with conditions A and B within the same runs; includes reference peaks
• 9: a small sphere inside a larger sphere (3D)

The addShape() function is provided for convenience when generating the pixelData for simulateImage(), as a simple way of adding morphological substructures using basic shapes such as squares and circles.

Value

For simulateSpectra, a list with elements:

• mz: a numeric vector of the observed m/z values
• intensity: a numeric vector or matrix of the intensities
For `simulateImage`, a `MSImagingExperiment` object. 
For `addShape`, a new `PositionDataFrame` with a logical column added for the corresponding shape. 
For `presetImageDef`, a list with two elements: the `pixelData` and `featureData` to be used as input to `simulateImage()`.

Author(s)

Kylie A. Bemis

See Also

`simspec`

Examples

```r
set.seed(1)

# generate a spectrum
s <- simulateSpectra(1)
plot(s$intensity ~ s$mz, type="l")

# generate a noisy low-resolution spectrum with a baseline
s <- simulateSpectra(1, baseline=2, sdnoise=0.3, resolution=100)
plot(s$intensity ~ s$mz, type="l")

# generate a high-resolution spectrum
s <- simulateSpectra(1, npeaks=100, resolution=10000)
plot(s$intensity ~ s$mz, type="l")

# generate an image
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))
peaks <- mz(metadata(mse)$design$featureData)
image(mse, mz=peaks[c(1,4,5,6)])
plot(mse, coord=c(x=3,y=3))
```

Description

Slice a spectral imaging dataset as a "data cube".

Usage

```r
slice(x, i = features(x, ...), ..., run = NULL,
      simplify = TRUE, drop = TRUE)
```
smooth

Arguments

x A spectral imaging dataset.
i The indices of features to slice for the images.
... Conditions describing features to slice, passed to features().
run The names of experimental runs to include, or the index of the levels of the runs to include.
simplify The image slices be returned as a list, or simplified to an array?
drop Should redundant array dimensions be dropped? If TRUE, dimensions with only one level are dropped using drop.

Value

A list or array of the sliced image(s). If multiple images are sliced and simplify=TRUE, then the last dimension will be the features.

Author(s)

Kylie A. Bemis

Examples

set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10), representation="centroid")
peaks <- mz(metadata(mse)$design$featureData)

# slice image for first feature
slice(mse, 1)

# slice by m/z-value
slice(mse, mz=peaks[1])

# slice multiple
slice(mse, mz=peaks[1:3])

smooth Smooth spectra

Description

Apply deferred smoothing to spectra.

Usage

## S4 method for signature 'SpectralImagingData'
smooth(x, method = c("gaussian", "bilateral", "adaptive", "diff", "guide", "pag", "sgolay", "ma"), ...)
Arguments

- **x**: A spectral imaging dataset.
- **method**: The smoothing method to use. See *filt1* for details.
- **...**: Additional arguments passed to the smoothing function.

Details

The supported smoothing methods are:

- "gaussian": Gaussian smoothing using *filt1_gauss*.
- "bilateral": Bilateral filter using *filt1_bi*.
- "adaptive": Adaptive bilateral filter using *filt1_adapt*.
- "diff": Nonlinear diffusion smoothing using *filt1_diff*.
- "guide": Guided filter using *filt1_guide*.
- "pag": Peak-aware guided filter using *filt1_pag*.
- "sgolay": Savitzky-Golar filter using *filt1_sg*.
- "ma": Moving average filter using *filt1_ma*.

Value

An object of the same class with the processing step queued.

Note

The smoothing is deferred until `process()` is called.

Author(s)

Kylie A. Bemis

See Also

*normalize, recalibrate, reduceBaseline, peakPick, process*

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(3,3))

# queue smoothing
mse2 <- smooth(mse, method="gaussian", width=11)
plot(mse2, i=4)

# apply smoothing
mse2 <- process(mse2)
```
Description

Apply cross-validation with an existing or a user-specified modeling function over folds of a spectral imaging dataset.

Usage

crossValidate(fit., x, y, folds = run(x), ..., predict. = predict, keep.models = FALSE, trainProcess = peakProcess, trainArgs = list(), testProcess = peakProcess, testArgs = list(), nchunks = getCardinalNChunks(), verbose = getCardinalVerbose(), BPPARAM = getCardinalBPPARAM())

## S4 method for signature 'SpatialCV'
fitted(object, type = c("response", "class"), ...)

## S4 method for signature 'SpatialCV'
image(x, i = 1L, type = c("response", "class"), layout = NULL, free = "", ...)

Arguments

fit. The function used to fit the model.
x, y The data and response variable, where x is assumed to be an P x N dataset such as a SpectralImagingExperiment
folds A vector coercible to a factor giving the fold for each row or column of x.
... Additional arguments passed to fit. and predict..
predict. The function used to predict on new data from the fitted model. The fitted model is passed as the 1st argument and the test data is passed as the 2nd argument.
keep.models Should the models be kept and returned?
trainProcess, trainArgs A function and arguments used for processing the training sets. The training set is passed as the 1st argument to trainProcess.
testProcess, testArgs A function and arguments used for processing the test sets. The test set is passed as the 1st argument to trainProcess, and the processed training set is passed as the 2nd argument.
verbose Should progress be printed for each iteration?
nchunks The number of chunks to use. Passed to fit., predict., trainProcess and testProcess.
**BPPARAM**  

**object**  
An object inheriting from SpatialCV.

**type**  
The type of prediction, where "response" means the fitted response matrix and "class" will be the vector of class predictions (only for classification).

**i**  
If predictions are made for multiple sets of parameters, which set of parameters (i.e., which element of the `fitted.values` list) should be plotted?

**layout**  
A vector of the form `c(nrow, ncol)` specifying the number of rows and columns in the facet grid.

**free**  
A string specifying the free spatial dimensions during faceting. E.g., "", "x", "y", "xy", "yx".

### Details

This method is designed to be used with the provided classification methods, but can also be used with user-provided functions and methods as long as they conform to certain expectations. Internally, `cv_do` from the `matter` package is used to perform the cross-validation. See `?cv_do` for details.

### Value

An object of class `SpatialCV` derived from `SpatialResults` and containing accuracies for each fold, the predictions for each fold, and (optionally) the fitted models.

### Author(s)

Kylie A. Bemis

### See Also

`cv_do`, `spatialShrunkenCentroids`, `PLS`, `OPLS`

---

**SpatialDGMM**  
*Spatially-aware Dirichlet Gaussian mixture model*

### Description

Fit a spatially-aware Gaussian mixture models to each feature. The model uses Dirichlet prior is used to achieve spatial smoothing. The means and standard deviations of the Gaussian components are estimated using gradient descent. Simulated annealing is used to avoid local optimia and achieve better parameter estimates.
Usage

```r
# S4 method for signature 'ANY'
spatialDGMM(x, coord, i, r = 1, k = 2, groups = NULL,
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(coord, r=r, groups=groups),
annealing = TRUE, compress = TRUE, byrow = FALSE,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

# S4 method for signature 'SpectralImagingExperiment'
spatialDGMM(x, i, r = 1, k = 2, groups = run(x),
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(coord(x), r=r, groups=groups), ...)

# S4 method for signature 'SpatialDGMM'
logLik(object, ...)

# S4 method for signature 'SpatialDGMM,missing'
plot(x, i = 1L, type = "density",
layout = NULL, free = "", ...)

# S4 method for signature 'SpatialDGMM'
image(x, i = 1L, type = "class",
layout = NULL, free = "", ...)
```

Arguments

- `x`: A spatial dataset in P x N matrix format.
- `i`: The rows/columns of `x` to segment (if not all of them).
- `coord`: The spatial coordinates of the rows/columns of `x`. Ignored if `neighbors` is provided.
- `r`: The spatial maximum distance for an observation to be considered a neighbor. Ignored if `neighbors` is provided.
- `k`: The number of Gaussian components.
- `groups`: Observations belonging to the different groups will be segmented independently. This should be set to the samples if statistic testing (via `meansTest` is to be performed.)
- `weights`: The type of spatial weights to use for the smoothing. Gaussian weights are weighted only by distance, while adaptive weights also consider the dissimilarity between neighboring observations.
- `neighbors`: A factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.
- `annealing`: Should simulated annealing be used?
- `compress`: Should the results be compressed? The results can be larger than the original dataset, so compressing them is useful. If this option is used, then the class
probabilities are not returned, and the class assignments are compressed using drle.

byrow Should the rows or columns of x be segmented?
nchunks The number of chunks to use for processing.
verbose Should progress messages be printed?
BPPARAM An optional instance of BiocParallelParam. See documentation for bplapply.
... Additional arguments passed to the next method.
object A SpatialDGMM object.
type The type of plot to display.
layout A vector of the form c(nrow, ncol) specifying the number of rows and columns in the facet grid.
free A string specifying the free spatial dimensions during faceting. E.g., "", "x", "y", "xy", "yx".

Value

An object of class SpatialDGMM derived from SpatialResults, containing the fitted sgmixn object and the spatial metadata.

Author(s)

Dan Guo and Kylie A. Bemis

References


Examples

set.seed(1)
mse <- simulateImage(preset=3, dim=c(10,10), npeaks=9,
                     peakheight=c(3,6,9), representation="centroid")

            gmm <- spatialDGMM(mse, r=1, k=4, weights="adaptive")

            image(gmm, i=1:9)
Description

Compute spatially-aware FastMap projection.

Usage

```r
## S4 method for signature 'ANY'
spatialFastmap(x, coord, r = 1, ncomp = 3,
               weights = c("gaussian", "adaptive"),
               neighbors = findNeighbors(coord, r=r),
               transpose = TRUE, niter = 3L,
               nchunks = getCardinalNChunks(),
               verbose = getCardinalVerbose(),
               BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
spatialFastmap(x, r = 1, ncomp = 3,
               weights = c("gaussian", "adaptive"),
               neighbors = findNeighbors(x, r=r), ...)

## S4 method for signature 'SpatialFastmap'
predict(object, newdata,
         neighbors = findNeighbors(newdata, r=object$r),
         nchunks = getCardinalNChunks(),
         BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpatialFastmap,missing'
plot(x, type = c("scree", "x"), ..., xlab, ylab)

## S4 method for signature 'SpatialFastmap'
image(x, type = "x", ...)
```

Arguments

- **x**: A spatial dataset in P x N matrix format.
- **coord**: The spatial coordinates of the rows/columns of `x`. Ignored if `neighbors` is provided.
- **r**: The spatial maximum distance for an observation to be considered a neighbor. Ignored if `neighbors` is provided.
- **ncomp**: The number of FastMap components.
- **weights**: The type of spatial weights to use for the smoothing. Gaussian weights are weighted only by distance, while adaptive weights also consider the dissimilarity between neighboring observations.
neighbors  A factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.

transpose  Should x be considered P x N?

niter  The number of iterations used to calculate the pivots for each FastMap component.

nchunks  The number of chunks to use for processing.

verbose  Should progress messages be printed?

BPPARAM  An optional instance of BiocParallelParam. See documentation for bplapply.

...  Additional arguments passed to the next method.

object  A SpatialFastmap object.

newdata  A new SpectralImagingExperiment for which to calculate the scores.

type  The type of plot to display.

xlab, ylab  Plotting labels.

Value

An object of class SpatialFastmap derived from SpatialResults, containing the fitted fastmap object and the spatial metadata.

Author(s)

Kylie A. Bemis

References


See Also

PCA, NMF, spatialKMeans

Examples

```r
set.seed(1)
mse <- simulateImage(preset=2, npeaks=20, dim=c(10,10), representation="centroid")

# project to FastMap components
fm <- spatialFastmap(mse, r=1, ncomp=2, weights="adaptive")

# visualize first 2 components
image(fm)
```
SpatialKMeans

Spatially-aware K-means clustering

Description

Perform spatially-aware k-means clustering. First the data is projected to a reduced dimension space using \texttt{spatialFastmap}. Then ordinary k-means clustering is applied to the projected data.

Usage

```r
## S4 method for signature 'ANY'
spatialKMeans(x, coord, r = 1, k = 2, ncomp = max(k),
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(coord, r=r),
transpose = TRUE, niter = 2L,
centers = TRUE, correlation = TRUE,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
spatialKMeans(x, r = 1, k = 2, ncomp = max(k),
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(x, r=r), ...)

## S4 method for signature 'SpatialKMeans'
topFeatures(object, n = Inf, sort.by = "correlation", ...)

## S4 method for signature 'SpatialKMeans,missing'
plot(x, type = c("correlation", "centers"), ..., xlab, ylab)

## S4 method for signature 'SpatialKMeans'
image(x, type = "cluster", ...)
```

Arguments

- **x**: A spatial dataset in P x N matrix format.
- **coord**: The spatial coordinates of the rows/columns of x. Ignored if neighbors is provided.
- **r**: The spatial maximum distance for an observation to be considered a neighbor. Ignored if neighbors is provided.
- **k**: The number of clusters.
- **ncomp**: The number of FastMap components.
- **weights**: The type of spatial weights to use for the smoothing. Gaussian weights are weighted only by distance, while adaptive weights also consider the dissimilarity between neighboring observations.
neighbors   A factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.
transpose   Should x be considered P x N?
niter       The number of iterations used to calculate the pivots for each FastMap component.
centers     Should the cluster centers be re-calculated on the original data?
correlation Should the correlations between features and the clusters be calculated?
ncunks      The number of chunks to use for processing.
verbose     Should progress messages be printed?
BPPARAM    An optional instance of BiocParallelParam. See documentation for bplapply.
...         Additional arguments passed to the next method.
object      A SpatialKMeans object.
n, sort.by  For topFeatures, the number of top features to return and how to sort them.
type        The type of plot to display.
xlab, ylab  Plotting labels.

Value
An object of class SpatialKMeans derived from SpatialResults, containing the fitted kmeans object and the spatial metadata.

Author(s)
Kylie A. Bemis

References

See Also
spatialKMeans spatialShrunkenCentroids

Examples
set.seed(1)
mse <- simulateImage(preset=3, dim=c(10,10), npeaks=20,
                     peakheight=c(3,6,9), representation="centroid")

# fit spatial k-means
skm <- spatialKMeans(mse, r=1, k=4, weights="adaptive")

# visualize clusters
image(skm)
SpatialNMF

Non-negative matrix factorization

Description

Compute nonnegative matrix factorization using alternating least squares or multiplicative updates.

Usage

```r
## S4 method for signature 'ANY'
NMF(x, ncomp = 3, method = c("als", "mult"),
    verbose = getCardinalVerbose(), ...)

## S4 method for signature 'SpectralImagingExperiment'
NMF(x, ncomp = 3, method = c("als", "mult"), ...)

## S4 method for signature 'SpatialNMF'
predict(object, newdata, ...)

## S4 method for signature 'SpatialNMF,missing'
plot(x, type = c("activation", "x"), ..., xlab, ylab)

## S4 method for signature 'SpatialNMF'
image(x, type = "x", ...)
```

Arguments

- `x`: A dataset in P x N matrix format.
- `ncomp`: The number of components to calculate.
- `method`: The method to use. Alternating least squares ("als") tends to be faster and potentially more accurate, but can be numerically unstable for data with high correlated features. Multiplicative updates ("mult") can be slower, but is more numerically stable.
- `verbose`: Should progress messages be printed?
- `...`: Options passed to `irlba`.
- `object`: A SpatialNMF object.
- `newdata`: A new SpectralImagingExperiment for which to calculate the scores.
- `type`: The type of plot to display.
- `xlab`, `ylab`: Plotting labels.

Value

An object of class SpatialNMF derived from SpatialResults, containing the fitted nnmf object and the spatial metadata.
Author(s)

Kylie A. Bemis

See Also

nnmf_als, nnmf_mult, PCA, spatialFastmap

Examples

```r
set.seed(1)
mse <- simulateImage(preset=2, npeaks=20, dim=c(10,10),
                      representation="centroid")

# project to principal components
mf <- NMF(mse, ncomp=2)

# visualize first 2 components
image(mf, superpose=FALSE, scale=TRUE)
```

SpatialPCA

Principal components analysis

Description

Compute principal components efficiently using implicitly restarted Lanczos bi-diagonalization (IRLBA) algorithm for approximate singular value decomposition.

Usage

```r
## S4 method for signature 'ANY'
PCA(x, ncomp = 3,
    center = TRUE, scale = FALSE,
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
PCA(x, ncomp = 3,
    center = TRUE, scale = FALSE, ...)

## S4 method for signature 'SpatialPCA'
predict(object, newdata,
         nchunks = getCardinalNChunks(),
         BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpatialPCA,missing'
plot(x, type = c("rotation", "scree", "x"), ..., xlab, ylab)
```
## S4 method for signature 'SpatialPCA'
image(x, type = "x", ...)

### Arguments

- **x**: A dataset in P x N matrix format.
- **ncomp**: The number of principal components to calculate.
- **center**: Should the data be centered?
- **scale**: Should the data be scaled?
- **nchunks**: The number of chunks to use for processing.
- **verbose**: Should progress messages be printed?
- **BPPARAM**: An optional instance of BiocParallelParam. See documentation for `bplapply`.
- **...**: Options passed to `irlba`.
- **object**: A SpatialPCA object.
- **newdata**: A new SpectralImagingExperiment for which to calculate the scores.
- **type**: The type of plot to display.
- **xlab, ylab**: Plotting labels.

### Value

An object of class SpatialPCA derived from SpatialResults, containing the fitted `prcomp_lanczos` object and the spatial metadata.

### Author(s)

Kylie A. Bemis

### See Also

`prcomp_lanczos`, `NMF`, `spatialFastmap`, `irlba`, `svd`

### Examples

```r
set.seed(1)
mse <- simulateImage(preset=2, npeaks=20, dim=c(10,10),
                      representation="centroid")

# project to principal components
pc <- PCA(mse, ncomp=2)

# visualize first 2 components
image(pc, superpose=FALSE, scale=TRUE)
```
SpatialPLS

Partial least squares (projection to latent structures)

Description

Compute partial least squares (also called projection to latent structures or PLS). This will also perform discriminant analysis (PLS-DA) if the response is a factor. Orthogonal partial least squares options (O-PLS and O-PLS-DA) is also supported; in this case, O-PLS step is a pre-processing step to remove noise orthogonal to the response, before fitting a PLS model with a single component.

Usage

```r
## S4 method for signature 'ANY'
PLS(x, y, ncomp = 3,
    method = c("nipals", "simpls", "kernel1", "kernel2"),
    center = TRUE, scale = FALSE, bags = NULL,
    nchunks = getCardinalNChunks(),
    verbose = getCardinalVerbose(),
    BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
PLS(x, y, ncomp = 3,
    method = c("nipals", "simpls", "kernel1", "kernel2"),
    center = TRUE, scale = FALSE, ...

## S4 method for signature 'SpatialPLS'
fitted(object, type = c("response", "class"), ...)

## S4 method for signature 'SpatialPLS'
predict(object, newdata, ncomp,
        type = c("response", "class"), simplify = TRUE, ...)

## S4 method for signature 'SpatialPLS'
topFeatures(object, n = Inf, sort.by = c("vip", "coefficients"), ...)

## S4 method for signature 'SpatialPLS,missing'
plot(x, type = c("coefficients", "vip", "scores"), ..., xlab, ylab)

## S4 method for signature 'SpatialPLS'
image(x, type = c("response", "class"), ...)

## S4 method for signature 'ANY'
OPLS(x, y, ncomp = 3, retx = TRUE,
     center = TRUE, scale = FALSE, bags = NULL,
     nchunks = getCardinalNChunks(),
     verbose = getCardinalVerbose(),
     BPPARAM = getCardinalBPPARAM(), ...)
```
## S4 method for signature 'SpectralImagingExperiment'
OPLS(x, y, ncomp = 3, retx = FALSE,
    center = TRUE, scale = FALSE, ...)

## S4 method for signature 'SpatialOPLS'
coef(object, ...)

## S4 method for signature 'SpatialOPLS'
residuals(object, ...)

## S4 method for signature 'SpatialOPLS'
fitted(object, type = c("response", "class"), ...)

## S4 method for signature 'SpatialOPLS'
predict(object, newdata, ncomp,
    type = c("response", "class"), simplify = TRUE, ...)

## S4 method for signature 'SpatialOPLS'
topFeatures(object, n = Inf, sort.by = c("vip", "coefficients"), ...)

## S4 method for signature 'SpatialOPLS,missing'
plot(x, type = c("coefficients", "vip", "scores"), ..., xlab, ylab)

## S4 method for signature 'SpatialOPLS'
image(x, type = c("response", "class"), ...)

### Arguments

**x**
A dataset in P x N matrix format.

**y**
The response variable.

**ncomp**
The number of principal components to calculate.

**method**
The method used for calculating the principal components. See pls for details.

**center**
Should the data be centered?

**scale**
Should the data be scaled?

**bags**
Bags for multiple instance learning. If provided, then it is assumed all observations within a bag have the same label, and if a single observation is "positive" then all observations in the bag are "positive". Multiple instance learning is performed using mi_learn.

**retx**
Should the (potentially large) processed data matrix be included in the result?

**nchunks**
The number of chunks to use for processing.

**verbose**
Should progress messages be printed?

**BPPARAM**
An optional instance of BiocParallelParam. See documentation for bplapply.

**...**
Options passed to irlba.

**object**
A SpatialPLS or SpatialOPLS object.
newdata: A new SpectralImagingExperiment for which to make predictions.

type: The type of fitted values to extract or the type of predictions to make.

simplify: If predictions are made using multiple numbers of components, should they be returned as a list, or simplified to an array?

n, sort.by: For topFeatures, the number of top features to return and how to sort them.

xlab, ylab: Plotting labels.

Value

An object of class SpatialPLS or SpatialOPLS derived from SpatialResults, containing the fitted pls or opls model and the spatial metadata.

Author(s)

Kylie A. Bemis

References


See Also

PCA, spatialShrunkenCentroids.

Examples

```r
set.seed(1)
msen <- simulateImage(preset=2, nppeaks=20, dim=c(10,10), representation="centroid")
cls <- makeFactor(circle=pData(msen)$circle, square=pData(msen)$square)

# fit a PLS model with 3 components
plsen <- PLS(msen, cls, ncomp=1:3)
plot(plsen, type="coefficients", annPeaks="circle")

# visualize predictions
image(plsen)
```

SpatialResults-class

The SpatialResults class provides a container for modeling results with spatial metadata. Most modeling functions applied to a SpectralImagingExperiment will return a SpatialResults-derived model object.
SpatialResults-class

Usage

## Instance creation
SpatialResults(model, data,
    featureData = if (!missing(data)) fData(data) else NULL,
    pixelData = if (!missing(data)) pData(data) else NULL)

## S4 method for signature 'SpatialResults,ANY'
plot(x, y, ..., 
    select = NULL, groups = NULL,
    superpose = TRUE, reducedDims = FALSE)

## S4 method for signature 'SpatialResults'
image(x, y, ..., 
    select = NULL, subset = TRUE,
    superpose = TRUE)

## Additional methods documented below

Arguments

- **model**
  The model object.
- **data**
  An object (typically the original dataset) with featureData and pixelData components.
- **featureData**
  A DataFrame with feature metadata, with a row for each feature.
- **pixelData**
  A PositionDataFrame with pixel metadata, with a row for each spectrum.
- **x, y**
  The model object and results to plot. (Not typically called directly.)
- **...**
  Additional options passed to plotting methods.
- **select**
  Select elements of the results to plot. For example, this selects a subset of matrix columns or a subset of factor levels to plot.
- **subset**
  A logical vector indicating which pixels to include in the image.
- **groups**
  A vector coercible to a factor indicating which of the specified spectra should be plotted with the same color.
- **superpose**
  If multiple results are plotted, should they be superposed on top of each other, or plotted separately?
- **reducedDims**
  Does this results component represent reduced dimensions (e.g., from PCA)?

Slots

- **model**: The model.
- **featureData**: A DataFrame containing feature-level metadata (e.g., a color channel, a molecular analyte, or a mass-to-charge ratio).
- **pixelData**: A PositionDataFrame containing spatial metadata, including each observation’s pixel coordinates and experimental run information.
Methods

modelData(object), modelData(object) <- value: Get or set the model slot.
featureData(object), featureData(object) <- value: Get or set the featureData slot.
fData(object), fData(object) <- value: Get or set the featureData slot.
featureNames(object), featureNames(object) <- value: Get or set the feature names (i.e.,
the row names of the featureData slot).
pixelData(object), pixelData(object) <- value: Get or set the elementMetadata slot.
pData(object), pData(object) <- value: Get or set the elementMetadata slot.
pixelNames(object), pixelNames(object) <- value: Get or set the pixel names (i.e., the row
names of the elementMetadata slot).
coord(object), coord(object) <- value: Get or set the pixel coordinate columns in pixelData.
coordNames(object), coordNames(object) <- value: Get or set the names of the pixel coordi-
nate columns in pixelData.
run(object), run(object) <- value: Get or set the experimental run column from pixelData.
runNames(object), runNames(object) <- value: Get or set the experimental run levels from
pixelData.
nrun(object): Get the number of experimental runs.

Author(s)

Kylie A. Bemis

See Also

ResultsList

SpatialShrunkenCentroids

Spatially-aware shrunken centroid clustering and classification

Description

Perform spatially-aware nearest shrunken centroid clustering or classification. These methods use
statistical regularization to shrink the t-statistics of the features toward 0 so that unimportant features
are removed from the model. The dissimilarity to class centroids are spatially smoothed.

Usage

## S4 method for signature 'ANY,ANY'
spatialShrunkenCentroids(x, y, coord, r = 1, s = 0,
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(coord, r=r), bags = NULL,
priors = table(y), center = NULL, transpose = TRUE,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment,ANY'
spatialShrunkenCentroids(x, y, r = 1, s = 0,
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(x, r=r), ...)

## S4 method for signature 'ANY,missing'
spatialShrunkenCentroids(x, coord, r = 1, k = 2, s = 0,
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(coord, r=r),
init = NULL, threshold = 0.01, niter = 10L,
center = NULL, transpose = FALSE,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment,missing'
spatialShrunkenCentroids(x, r = 1, k = 2, s = 0,
weights = c("gaussian", "adaptive"),
neighbors = findNeighbors(x, r=r), ...)

## S4 method for signature 'SpatialShrunkenCentroids'
fitted(object, type = c("response", "class"), ...)

## S4 method for signature 'SpatialShrunkenCentroids'
predict(object, newdata,
type = c("response", "class"),
neighbors = findNeighbors(newdata, r=object$r),
nchunks = getCardinalNChunks(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpatialShrunkenCentroids'
logLik(object, ...)

## S4 method for signature 'SpatialShrunkenCentroids'
topFeatures(object, n = Inf, sort.by = c("statistic", "centers"), ...)

## S4 method for signature 'SpatialShrunkenCentroids,missing'
plot(x, type = c("statistic", "centers"), ..., xlab, ylab)

## S4 method for signature 'SpatialShrunkenCentroids'
image(x, type = c("probability", "class"), ...)

Arguments

x                A spatial dataset in P x N matrix format.
y
The response variable.

coord
The spatial coordinates of the rows/columns of \( x \). Ignored if \( \text{neighbors} \) is provided.

r
The spatial maximum distance for an observation to be considered a neighbor. Ignored if \( \text{neighbors} \) is provided.

k
The number of classes for clustering.

s
The sparsity parameter.

weights
The type of spatial weights to use for the smoothing. Gaussian weights are weighted only by distance, while adaptive weights also consider the dissimilarity between neighboring observations.

neighbors
A factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.

bags
Bags for multiple instance learning. If provided, then it is assumed all observations within a bag have the same label, and if a single observation is "positive" then all observations in the bag are "positive". Multiple instance learning is performed using \texttt{mi.learn}.

priors
The (unnormalized) prior probabilities for each class.

center
The global centroid (if known).

transpose
Should \( x \) be considered \( P \times N \)?

init
A list of initial cluster configurations. (Should resemble the output of \texttt{kmeans}.)

threshold
Stop iteration when the proportion of cluster assignment updates is less than this threshold.

niter
The maximum number of iterations.

nchunks
The number of chunks to use for processing.

verbose
Should progress messages be printed?

BPPARAM
An optional instance of \texttt{BiocParallelParam}. See documentation for \texttt{bplapply}.

...
Additional arguments passed to the next method.

object
A \texttt{SpatialShrunkenCentroids} object.

cndata
A new \texttt{SpectralImagingExperiment} for which to make predictions.

type
The type of fitted values to extract or the type of predictions to make.

n, sort.by
For \texttt{topFeatures}, the number of top features to return and how to sort them.

xlab, ylab
Plotting labels.

Value
An object of class \texttt{SpatialShrunkenCentroids} derived from \texttt{SpatialResults}, containing the fitted \texttt{nscentroids} object and the spatial metadata.

Author(s)
Kylie A. Bemis
spatialWeights

Calculate spatial weights

Description

Calculate weights for neighboring observations based on either the spatial distance between the neighbors or the dissimilarity between the observations.

Usage

```r
## S4 method for signature 'ANY'
spatialWeights(x, coord = x, r = 1, byrow = TRUE,
               neighbors = findNeighbors(coord, r=r),
               weights = c("gaussian", "adaptive"),
               sd = ((2 * r) + 1) / 4, matrix = FALSE,
               nchunks = getCardinalNChunks(),
               verbose = getCardinalVerbose(),
               BPPPARAM = getCardinalBPPARAM(), ...)
```

References


See Also

spatialKMeans

Examples

```r
set.seed(1)
mse <- simulateImage(preset=3, dim=c(10,10), npeaks=20,
                     peakheight=c(3,6,9), representation="centroid")

# fit spatial shrunken centroids
ssc <- spatialShrunkenCentroids(mse, r=1, k=4, s=c(0,3,6,9), weights="adaptive")

# visualize classes
image(ssc, i=1:4)

# visualize t-statistics
plot(ssc, i=1:4)
```
### spatialWeights

#### Arguments

- **x**
  - Either matrix or data frame of spatial coordinates, or a data matrix with rows or columns located at the coordinates given by `coord`.

- **coord**
  - The spatial coordinates of the rows/columns of `x`. Ignored if `neighbors` is provided.

- **r**
  - The spatial maximum distance for an observation to be considered a neighbor. Ignored if `neighbors` is provided.

- **byrow**
  - If `x` is a data matrix, then are the weights calculated based on the dissimilarity between the rows (TRUE) or the columns (FALSE).

- **neighbors**
  - A factor giving which observations should be treated as spatially-independent. Observations in the same group are assumed to have a spatial relationship.

- **weights**
  - The type of weights to calculate. Either Gaussian weights with a constant standard deviation, or adaptive weights with a standard deviation based on the dissimilarity between the neighboring observations.

- **sd**
  - The standard deviation for the Gaussian weights. Ignored with `weights="adaptive"`.

- **matrix**
  - Should the weights be returned as a sparse adjacency matrix instead of a list?

- **nchunks**
  - The number of chunks to use for processing.

- **verbose**
  - Should progress messages be printed?

- **BPPARAM**
  - An optional instance of `BiocParallelParam`. See documentation for `bplapply`.

- **...**
  - Additional arguments passed to the next method.

#### Value

Either a list of weights of neighbors or a sparse adjacency matrix (`sparse_mat`).

#### Author(s)

Kylie A. Bemis

#### See Also

- `findNeighbors`
Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, dim=c(10,10))

# calculate weights based on distance
spatialWeights(pixelData(mse), r=1)

# calculate weights based on spectral dissimilarity
spatialWeights(mse, r=1)
```

Description

The `SpectraArrays` class provides a list-like container for spectra arrays of conformable dimensions.

Usage

```r
## Instance creation
SpectraArrays(arrays = SimpleList())
```

### Additional methods documented below

Arguments

- `arrays`: A list of arrays.

Details

The `SpectraArrays` class is intended to be flexible and the arrays do not need to be "array-like" (i.e., have non-NULL `dim()`.) One dimensional arrays and lists are allowed. Every array must have the same `NROW()` and `NCOL()`.

It supports lossless coercion to and from `SimpleList`.

Methods

All methods for `SpectralImagingData` also work on `SpectraArrays` objects. Additional methods are documented below:

- `length(object)`: Get the number of spectra in the object.
- `names(object), names(object) <- value`: Get or set the names of spectra arrays in the object.
- `object[[i]], object[[i]] <- value`: Get or set an array in the object.
- `object[i, j, ..., drop]`: Subset as a list or array, depending on the number of dimensions of the stored spectra arrays. The result is the same class as the original object.
- `rbind(...), cbind(...): Combine SpectraArrays objects by row or column.
- `c(...): Combine SpectraArrays objects as lists.`
SpectralImagingArrays-class

Author(s)
Kylie A. Bemis

See Also
SpectralImagingData, MSImagingArrays

Examples

```r
set.seed(1)
x <- matrix(rlnorm(128), nrow=16, ncol=8)
y <- matrix(rlnorm(128), nrow=16, ncol=8)

s <- SpectraArrays(list(x=x, y=y))
print(s)
```

Description

The `SpectralImagingArrays` class provides a list-like container for high-dimensional spectral imaging data where every spectrum may have its own domain values. It is designed to provide easy access to raw individual spectra, but images cannot be easily reconstructed.

The `MSImagingArrays` class extends `SpectralImagingArrays` for mass spectrometry-based imaging experiments with unaligned mass features.

Usage

```r
## Instance creation
SpectralImagingArrays(spectraData = SimpleList(),
pixelData = PositionDataFrame(), metadata = list())
```

## Additional methods documented below

Arguments

- `spectraData`: Either a list-like object with lists of individual spectra and lists of their domain values, or a `SpectraArrays` instance.
- `pixelData`: A `PositionDataFrame` with pixel metadata, with a row for each spectrum.
- `metadata`: A list with experimental-level metadata.
Slots

spectraData: A SpectraArrays object storing one or more array-like data elements with conformable dimensions.

elementMetadata: A PositionDataFrame containing spectrum-level metadata, including each spectrum’s pixel coordinates and experimental run information.

processing: A list containing unexecuted ProcessingStep objects.

Methods

All methods for SpectralImagingData also work on SpectralImagingArrays objects. Additional methods are documented below:

length(object): Get the number of spectra in the object.

object[i, ..., drop]: Subset as a list based on the spectra. The result is the same class as the original object.

rbind(...), cbind(...): Combine SpectralImagingArrays objects by row or column.

Author(s)

Kylie A. Bemis

See Also

SpectralImagingData, MSImagingArrays

Examples

set.seed(1)
x <- replicate(9, rlnorm(10), simplify=FALSE)
t <- replicate(9, sort(runif(10)), simplify=FALSE)
coord <- expand.grid(x=1:3, y=1:3)

sa <- SpectralImagingArrays(
    spectraData=list(intensity=x, wavelength=t),
    pixelData=PositionDataFrame(coord))

print(sa)
Description

The SpectralImagingData class is an abstract container for high-dimensional spectral imaging data. Every spectrum is associated with spatial coordinates so that an image can be constructed from the spectral intensities.

The SpectralImagingArrays and SpectralImagingExperiment classes directly extend this class, where SpectralImagingArrays is primarily intended for unprocessed spectra with unaligned features, and SpectralImagingExperiment is intended for processed spectra with aligned features.

The MSImagingArrays and MSImagingExperiment classes further extend these classes for mass spectrometry imaging data.

Slots

spectraData: A SpectraArrays object storing one or more array-like data elements with conformable dimensions.

elementMetadata: A PositionDataFrame containing spectrum-level metadata, including each spectrum’s pixel coordinates and experimental run information.

processing: A list containing unexecuted ProcessingStep objects.

Methods

spectraData(object, ...), spectraData(object, ...) <- value: Get or set the spectraData slot.
spectraNames(object, ...), spectraNames(object, ...) <- value: Get or set the names of the spectra in the spectraData slot.
spectra(object, i = 1L, ...), spectra(object, i = 1L, ...) <- value: Get or set a specific spectra array in the spectraData slot.
pixelData(object), pixelData(object) <- value: Get or set the elementMetadata slot.
pData(object), pData(object) <- value: Get or set the elementMetadata slot.
pixelNames(object), pixelNames(object) <- value: Get or set the pixel names (i.e., the row names of the elementMetadata slot).
spectraVariables(object, ...): Get the names of the spectrum-level variables (i.e., the columns of the elementMetadata slot).
coord(object), coord(object) <- value: Get or set the pixel coordinate columns in pixelData.
coordNames(object), coordNames(object) <- value: Get or set the names of the pixel coordinate columns in pixelData.
run(object), run(object) <- value: Get or set the experimental run column from pixelData.
runNames(object), runNames(object) <- value: Get or set the experimental run levels from pixelData.
nrun(object): Get the number of experimental runs.
is3D(object): Check if the number of spatial dimensions is greater than 2.

processingData(object, ...), processingData(object, ...) <- value: Get or set the processing slot.
**SpectralImagingExperiment-class**

**Author(s)**
Kylie A. Bemis

**See Also**
SpectralImagingExperiment, SpectralImagingArrays, MSImagingExperiment, MSImagingArrays

---

**Description**

The `SpectralImagingExperiment` class provides a matrix-like container for high-dimensional spectral imaging data where every spectrum shares the same domain values. It is designed to provide easy access to both the spectra (as columns) and sliced images (as rows).

The `MSImagingExperiment` class extends SpectralImagingExperiment for mass spectrometry-based imaging experiments with aligned mass features.

**Usage**

```
## Instance creation
SpectralImagingExperiment(spectraData = SimpleList(),
   featureData = DataFrame(), pixelData = PositionDataFrame(),
   metadata = list())

## Additional methods documented below
```

**Arguments**

- `spectraData`: Either a matrix-like object with number of rows equal to the number of features and number of columns equal to the number of pixels, a list of such objects, or a `SpectraArrays` instance.
- `featureData`: A `DataFrame` with feature metadata, with a row for each feature.
- `pixelData`: A `PositionDataFrame` with pixel metadata, with a row for each spectrum.
- `metadata`: A list with experimental-level metadata.

**Slots**

- `spectraData`: A `SpectraArrays` object storing one or more array-like data elements with conformable dimensions.
- `featureData`: A `DataFrame` containing feature-level metadata (e.g., a color channel, a molecular analyte, or a mass-to-charge ratio).
- `elementMetadata`: A `PositionDataFrame` containing spectrum-level metadata, including each spectrum’s pixel coordinates and experimental run information.
- `processing`: A list containing unexecuted `ProcessingStep` objects.
Methods

All methods for `SpectralImagingData` also work on `SpectralImagingExperiment` objects. Additional methods are documented below:

- `featureData(object), featureData(object) <- value`: Get or set the `featureData` slot.
- `fData(object), fData(object) <- value`: Get or set the `featureData` slot.
- `featureNames(object), featureNames(object) <- value`: Get or set the feature names (i.e., the row names of the `featureData` slot).
- `length(object)`: Get the number of spectra in the object.
- `nrow(object), ncol(object)`: Get the number of rows (features) or the number of columns (pixels) in the object.
- `object[i, j, ..., drop]`: Subset based on the rows (`featureData`) and the columns (`pixelData`). The result is the same class as the original object.
- `rbind(...), cbind(...)`: Combine `SpectralImagingExperiment` objects by row or column.

Author(s)

Kylie A. Bemis

See Also

`SpectralImagingData, MSImagingExperiment`

Examples

```r
set.seed(1)
x <- matrix(rnorm(81), nrow=9, ncol=9)
index <- 1:9
coord <- expand.grid(x=1:3, y=1:3)

se <- SpectralImagingExperiment(
spectraData=x,
featureData=DataFrame(index=1:9),
pixelData=PositionDataFrame(coord))

print(se)
```

---

**spectrapply**  
*Apply functions over spectra*

Description

Apply a user-specified function over all spectra in a spectral imaging dataset.
spectrapply

Usage

## S4 method for signature 'MSImagingExperiment'
spectrapply(object, FUN, ...,
  spectra = "intensity", index = "mz")

## S4 method for signature 'MSImagingArrays'
spectrapply(object, FUN, ...,
  spectra = "intensity", index = "mz")

## S4 method for signature 'SpectralImagingExperiment'
spectrapply(object, FUN, ...,
  spectra = "intensity", index = NULL,
  simplify = TRUE, outpath = NULL,
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM())

## S4 method for signature 'SpectralImagingArrays'
spectrapply(object, FUN, ...,
  spectra = "intensity", index = NULL,
  simplify = TRUE, outpath = NULL,
  nchunks = getCardinalNChunks(),
  verbose = getCardinalVerbose(),
  BPPARAM = getCardinalBPPARAM())

Arguments

object A spectral imaging dataset.
FUN A function to be applied. The function must accept (1) a first argument giving
the spectral intensities as a numeric vector and (2) a second argument giving the
intensity locations (e.g., m/z values) as a numeric vector.
... Options passed to chunkMapply or chunkApply.
spectra The name of the array in spectraData() to use for the spectral intensities.
index The name of the array in spectraData() (for SpectralImagingArrays) or
column in featureData() (for SpectralImagingExperiment) to use for the
spectral locations.
simplify Should the result be simplified to an array if possible?
outpath Optional. The name of a file to write the resulting data.
nchunks The number of chunks to use for processing.
verbose Should progress messages be printed?
BPPARAM An optional instance of BiocParallelParam. See documentation for bplapply.

Value

A list if simplify=FALSE. Otherwise, a vector or matrix, or a higher-dimensional array if the at-
temted simplification is successful.
Author(s)
Kylie A. Bemis

See Also
summarizeFeatures, summarizePixels

Examples

```r
set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))

# find m/z locations of peaks in each spectrum
peaks <- spectrapply(mse, function(x, mz) mz[matter::findpeaks(x)]

head(peaks[[1L]])
head(peaks[[2L]])
```

subsetFeatures
Subset a spectral imaging dataset

Description
Returns a subset of the dataset that meets the conditions.

Usage

```r
## S4 method for signature 'SpectralImagingArrays'
subset(x, subset, ...)

## S4 method for signature 'SpectralImagingExperiment'
subset(x, select, subset, ...)
```

subsetFeatures(x, ...)
subsetPixels(x, ...)

Arguments

- `x` A spectral imaging dataset.
- `select` Logical expression to be evaluated in the object’s `featureData()` indicating which rows (features) to keep.
- `subset` Logical expression to be evaluated in the object’s `pixelData()` indicating which columns (pixels) to keep.
- `...` Conditions describing rows (features) or columns (pixels) to be retained. Passed to `features()` and `pixels()` methods to obtain the subset indices.
**summarizeFeatures**

**Value**

An object of the same class as `x` with the appropriate subsetting applied to it.

**Author(s)**

Kylie A. Bemis

**Examples**

```r
case.set.seed(1)
case mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))

# subset features to mass range 1000 - 1500
case subsetFeatures(mse, 1000 < mz, mz < 1500)

# select pixels to coordinates x = 1..3, y = 1..3
case subsetPixels(mse, x <= 3, y <= 3)

# subset both features + pixels
case subset(mse, 1000 < mz & mz < 1500, x <= 3 & y <= 3)
```

---

**summarizeFeatures**　*Summarize a spectral imaging dataset*

**Description**

Summarizes over the rows or columns of the dataset.

**Usage**

```r
summarizeFeatures(x, stat = "mean", groups = NULL,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

summarizePixels(x, stat = c(tic="sum"), groups = NULL,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM(), ...)

## S4 method for signature 'SpectralImagingExperiment'
rowStats(x, stat, ...,
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM())

## S4 method for signature 'SpectralImagingExperiment'
colStats(x, stat, ...)```
nchunks = getCardinalNChunks(),
verbose = getCardinalVerbose(),
BPPARAM = getCardinalBPPARAM())

## S4 method for signature 'SpectralImagingExperiment'
rowSums(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'SpectralImagingExperiment'
colSums(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'SpectralImagingExperiment'
rowMeans(x, na.rm = FALSE, dims = 1, ...)

## S4 method for signature 'SpectralImagingExperiment'
colMeans(x, na.rm = FALSE, dims = 1, ...)

Arguments

x            A spectral imaging dataset.
stat          The name of summary statistics to compute over the rows or columns of a matrix. Allowable values include: "min", "max", "prod", "sum", "mean", "var", "sd", "any", "all", and "nnzero".
groups        A vector coercible to a factor giving groups to summarize.
na.rm         If TRUE, remove NA values before summarizing.
dims          Ignored.
nchunks       The number of chunks to use for processing.
verbose       Should progress messages be printed?
BPPARAM       An optional instance of BiocParallelParam. See documentation for bplapply.
...            Additional arguments passed to rowStats or colStats, such as the number of chunks.

Value

For summarizeFeatures and summarizePixels, an object of the same class as x with the statistical summaries added as columns in the featureData() or pixelData(), respectively.

For rowStats, colStats, etc., a vector, matrix, or array with the summary statistics.

Author(s)

Kylie A. Bemis

Examples

set.seed(1)
mse <- simulateImage(preset=1, npeaks=10, dim=c(10,10))

# summarize mean spectrum
mse <- summarizeFeatures(mse, stat="mean")
plot(mse, "mean")

# summarize total ion current
mse <- summarizePixels(mse, stat=c(TIC="sum"))
image(mse, "TIC")

writeMSIData  Write mass spectrometry imaging data files

Description

Write supported mass spectrometry imaging data files, including imzML and Analyze 7.5.

Usage

writeMSIData(object, file, ...)

## S4 method for signature 'MSImagingExperiment_OR_Arrays'
writeImzML(object, file, bundle = TRUE,
verbose = getCardinalVerbose(), ...)

## S4 method for signature 'MSImagingExperiment'
writeAnalyze(object, file, verbose = getCardinalVerbose(), ...)

## S4 method for signature 'SpectralImagingExperiment'
writeAnalyze(object, file, verbose = getCardinalVerbose(), ...)

Arguments

object  A spectral imaging dataset.
file    The absolute or relative file path. The file extension must be included for writeMSIData.
bundle  Should the "imzML" and "ibd" files be bundled into a new directory of the same name?
verbose Should progress messages be printed?
...     Additional arguments passed to writeImzML or writeAnalyze.

Details

The writeImzML function supports writing both the "continuous" and "processed" formats. Exporting the experimental metadata to cvParam tags is lossy, and not all metadata will be preserved. If exporting an object that was originally imported from an imzML file, only metadata that appears in experimentData() will be preserved when writing.

Datasets with multiple experimental runs will be merged into a single file. The object’s pixelData() and featureData() will also be written to tab-delimited files if appropriate. These will be read back in by readImzML().

The imzML files can be modified after writing (such as to add additional experimental metadata) using the Java-based imzMLValidator application: https://gitlab.com/imzML/imzMLValidator/.
Value

TRUE if the file was written successfully, with the output file paths attached as an attribute.

Author(s)

Kylie A. Bemis

References


See Also

readMSIData

XDataFrame-class

XDataFrame: Extended data frame with key columns

Description

The XDataFrame extends the DataFrame class from the S4Vectors package with support for columns (or sets of columns) designated as keys.

Usage

XDataFrame(..., keys = list())

Arguments

... Arguments passed to the DataFrame().
keys A named list of character vectors giving the names of key columns. The names of the list become the names of the keys (which may be different from the columns). The character vectors specify the names of columns that compose that key.

Details

For the most part, XDataFrame behaves identically to DataFrame, and key columns can be get or set as usual. The XDataFrame class is primarily intended as a way to enforce additional requirements or constraints on specific sets of columns in a structured way. It provides an abstracted way of manipulating sets of columns that are expected to follow certain rules. The keys remain consistent and accessible even if the columns of the data frame are renamed.
The base class currently has only minimal requirements for keys (i.e., that they are valid columns in the data frame). Additionally, keys are checked for compatibility when combining data frames. Uniqueness is not checked.

Subclasses can enforce additional constraints on key columns. For example, the PositionDataFrame and MassDataFrame classes.

Methods

keys(object, i = NULL, ..., drop = TRUE), keys(object, i = NULL, ...) <- value: Get or set the key columns. By default, this gets or sets the keys slot. Provide i to get or set specific keys.

Author(s)

Kylie A. Bemis

See Also

DataFrame, MassDataFrame, PositionDataFrame

Examples

## Create an XDataFrame object
XDataFrame(id=1:10, letter=LETTERS[1:10], keys=list(index="id"))
**Index**

* **IO**
  - `readMSIData`, 32
  - `writeMSIData`, 75

* **classes**
  - `MassDataFrame-class`, 10
  - `MSImagingArrays-class`, 13
  - `MSImagingExperiment-class`, 15
  - `PositionDataFrame-class`, 29
  - `ResultsList-class`, 37
  - `SpectraArrays-class`, 65
  - `SpectralImagingArrays-class`, 66
  - `SpectralImagingData-class`, 67
  - `SpectralImagingExperiment-class`, 69
  - `XDataFrame-class`, 76

* **classif**
  - `SpatialCV`, 45
  - `SpatialPLS`, 56
  - `SpatialShrunkenCentroids`, 60

* **clustering**
  - `SpatialDGMM`, 46
  - `SpatialKMeans`, 51
  - `SpatialShrunkenCentroids`, 60

* **datagen**
  - `simulateSpectra`, 39

* **hplot**
  - `plot-image`, 25
  - `plot-spectra`, 27

* **htest**
  - `MeansTest`, 11

* **iplot**
  - `selectROI`, 38

* **manip**
  - `process`, 30
  - `slice`, 42
  - `spectrapply`, 70
  - `subsetFeatures`, 72

* **methods**
  - `colocalized`, 5

* **models**
  - `MeansTest`, 11

* **multivariate**
  - `SpatialFastmap`, 49
  - `SpatialNMF`, 53
  - `SpatialPCA`, 54
  - `SpatialPLS`, 56

* **package**
  - `Cardinal-package`, 3

* **regression**
  - `MeansTest`, 11
  - `SpatialCV`, 45

* **smooth**
  - `smooth`, 43

* **spatial**
  - `findNeighbors`, 9
  - `SpatialDGMM`, 46
  - `SpatialFastmap`, 49
  - `SpatialKMeans`, 51
  - `SpatialShrunkenCentroids`, 60
  - `spatialWeights`, 63

* **ts**
  - `bin`, 4
  - `estimateDomain`, 7
  - `normalize`, 17
  - `peakAlign`, 18
  - `peakPick`, 20
  - `peakProcess`, 22
  - `process`, 30
  - `recalibrate`, 34
  - `reduceBaseline`, 35
  - `smooth`, 43

* **univar**
  - `summarizeFeatures`, 73

* **utilities**
  - `estimateDomain`, 7
  - `features`, 8
  - `pixels`, 24
SpatialCV, 45
[M, MassDataFrame, ANY, ANY, ANY-method
(MassDataFrame-class), 10
[., PositionDataFrame, ANY, ANY, ANY-method
(PositionDataFrame-class), 29
[, SpectraArrays, ANY, ANY, ANY-method
(SpectraArrays-class), 65
[, SpectralImagingArrays, ANY, ANY, ANY-method
(SpectralImagingArrays-class), 66
[, SpectralImagingExperiment, ANY, ANY, ANY-method
(SpectralImagingExperiment-class), 69
[, XDataFrame, ANY, ANY, ANY-method
(XDataFrame-class), 76
<-, SpectraArrays, ANY, ANY, ANY-method
(SpectraArrays-class), 65
<-, SpectralImagingArrays, ANY, ANY, ANY-method
(SpectralImagingArrays-class), 66
<-, SpectralImagingExperiment, ANY, ANY, ANY-method
(SpectralImagingExperiment-class), 69
<-, XDataFrame, ANY, ANY, ANY-method
(XDataFrame-class), 76
[, SpatialResults-method
(SpatialResults-class), 58
[, SpectraArrays-method
(SpectraArrays-class), 65
[, SpectralImagingData-method
(SpectralImagingData-class), 67
<-, SpectraArrays-method
(SpectraArrays-class), 65
<-, SpectralImagingData-method
(SpectralImagingData-class), 67
<-, XDataFrame-method
(XDataFrame-class), 76
$, SpatialResults-method
(SpatialResults-class), 58
$, SpectraArrays-method
(SpectraArrays-class), 65
$, SpectralImagingData-method
(SpectralImagingData-class), 67
<-, SpectraArrays-method
(SpectraArrays-class), 65
<-, SpectralImagingArrays-method
(SpectralImagingArrays-class), 66
<-, SpectralImagingExperiment-method
(SpectralImagingExperiment-class), 69
<-, XDataFrame-method
(XDataFrame-class), 76
addProcessing, SpectralImagingData-method
(process), 30
addShape (simulateSpectra), 39
aggregate, SpectralImagingExperiment-method
(deprecated), 7
alpha.colors (deprecated), 7
approx1, 5
BaselineReduction (deprecated), 7
BaselineReduction<-, (deprecated), 7
bin, 4
bin, MSImagingArrays-method (bin), 4
bin, MSImagingExperiment-method (bin), 4
bin, SpectralImagingArrays-method (bin), 4
bin, SpectralImagingExperiment-method
(bin), 4
binpeaks, 19
bplapply, 3, 6, 8, 12, 19, 22, 31, 33, 40, 46,
48, 50, 52, 55, 57, 62, 64, 71, 74
bw.colors (deprecated), 7
c, MSImagingArrays-method
(MSImagingArrays-class), 13
c, SpectraArrays-method
(SpectraArrays-class), 65
c, SpectralImagingArrays-method
(SpectralImagingArrays-class), 66
Cardinal (Cardinal-package), 3
Cardinal-package, 3
CardinalVersion (Cardinal-package), 3
cbind, MSImagingExperiment-method
(MSImagingExperiment-class), 15
cbind, SpectraArrays-method
(SpectraArrays-class), 65
cbind, SpectralImagingArrays-method
(SpectralImagingArrays-class), 66
cbind, SpectralImagingExperiment-method
(SpectralImagingExperiment-class), 69
cbind, XDataFrame-method
(XDataFrame-class), 76
centroided, MSImagingExperiment_OR_Arrays-method
(MSImagingExperiment-class), 15
centroided<-,MSImagingExperiment_OR_Arrays-method
(MSImagingExperiment-class), 15
chunk_colapply, 31
chunk_mapply, 31
chunkApply, 6, 71
chunkLapply, 8
chunkMapply, 71
class:Hashmat (deprecated), 7
class:IAnnotatedDataFrame (deprecated), 7
class:ImageData (deprecated), 7
class:iSet (deprecated), 7
class:MassDataFrame (MassDataFrame-class), 10
class:MeansTest (MeansTest), 11
class:MIAPEReimage (deprecated), 7
class:MSImagingData (deprecated), 7
class:MSImageProcess (deprecated), 7
class:MSImageSet (deprecated), 7
class:MSImagingArrays (MSImagingArrays-class), 13
class:MSImagingExperiment (MSImagingExperiment-class), 15
class:PositionDataFrame (PositionDataFrame-class), 29
class:ResultSet (deprecated), 7
class:SimpleDataFrame (deprecated), 7
class:SpatialCV (SpatialCV), 45
class:SpatialDGMM (SpatialDGMM), 46
class:SpatialFastmap (SpatialFastmap), 49
class:SpatialKMeans (SpatialKMeans), 51
class:SpatialNMF (SpatialNMF), 53
class:SpatialOLPLS (SpatialOLPLS), 56
class:SpatialPCA (SpatialPCA), 54
class:SpatialPLS (SpatialPLS), 56
class:SpatialResults (SpatialResults-class), 58
class:SpatialShrunkenCentroids (SpatialShrunkenCentroids), 60
class:SpectraArrays (SpectraArrays-class), 65
class:SpectralImagingArrays (SpectralImagingArrays-class), 66

methods:SpectralImagingData
(SpectralImagingData-class), 67
class:SpectralImagingExperiment
(SpectralImagingExperiment-class), 69
class:XDataFrame (XDataFrame-class), 76
class:XFrame (XDataFrame-class), 76
class:NameForDisplay, XFrame-method
(XDataFrame-class), 76
coeff,SpatialPLS-method (SpatialPLS), 56
c coerce,DataFrame, XDataFrame-method
(XDataFrame-class), 76
c coerc,DFrame, MassDataFrame-method
(MassDataFrame-class), 10
c coerces,DFrame, PositionDataFrame-method
(PositionDataFrame-class), 29
c coerce, list, SpectraArrays-method
(SpectraArrays-class), 65
c coerce,MSImageSet, MSImagingExperiment-method
(MSImagingExperiment-class), 15
c coerce,MSImagingArrays, MSImagingExperiment-method
(readMSIData), 32
c coerc, MSImagingExperiment, MSImagingArrays-method
(readMSIData), 32
c coerc, SimpleList, SpectraArrays-method
(SpectraArrays-class), 65
c coerces, SpectraArrays, list-method
(SpectraArrays-class), 65
c coerces, SpectraArrays, SimpleList-method
(SpectraArrays-class), 65
col.map (deprecated), 7
colMeans, SpectralImagingExperiment-method
(summarizeFeatures), 73
colnames, SpectralImagingExperiment-method
(SpectralImagingExperiment-class), 69
colnames<-, SpectralImagingExperiment-method
(SpectralImagingExperiment-class), 69
colocalized, 5
colocalized, MSImagingExperiment-method
(colocalized), 5
colocalized, SpatialDGMM-method
(colocalized), 5
colocalized, SpectralImagingExperiment-method
(colocalized), 5
colStats, 74
fData<-, SpatialResults, ANY-method (SpatialResults-class), 58
fData<-, SpectralImagingExperiment, ANY-method (SpectralImagingExperiment-class), 69
featureApply (deprecated), 7
featureApply, SpectralImagingExperiment-method (deprecated), 7
featureData, SpatialResults-method (SpatialResults-class), 58
featureData, SpectralImagingExperiment-method (SpectralImagingExperiment-class), 69
featureData<-, SpatialResults, ANY-method (SpatialResults-class), 58
featureData<-, SpectralImagingExperiment, ANY-method (SpectralImagingExperiment-class), 69
featureNames, SpatialResults-method (SpatialResults-class), 58
featureNames, SpectralImagingExperiment-method (SpectralImagingExperiment-class), 69
featureNames<-, SpatialResults-method (SpatialResults-class), 58
featureNames<-, SpectralImagingExperiment-method (SpectralImagingExperiment-class), 69
features, 8
features, MSImagingExperiment-method (features), 8
features, SpectralImagingExperiment-method (features), 8
filt1, 44
filt1_adapt, 44
filt1_bi, 44
filt1_diff, 44
filt1_gauss, 44
filt1_guide, 44
filt1_ma, 44
filt1_pag, 44
filt1_sg, 44
filt2, 26
findNeighbors, 9, 64
findNeighbors, ANY-method (findNeighbors), 9
findNeighbors, PositionDataFrame-method (findNeighbors), 9
findNeighbors, SpectralImagingData-method (findNeighbors), 9
findpeaks, 8, 20, 22
findpeaks_cwt, 21
fitted, ResultsList-method (ResultsList-class), 37
fitted, SpatialCV-method (SpatialCV), 45
fitted, SpatialOPLS-method (SpatialOPLS), 56
fitted, SpatialPLS-method (SpatialPLS), 56
fitted, SpatialResults-method (SpatialResults-class), 58
fitted, SpatialShrunkenCentroids-method (SpatialShrunkenCentroids), 60
getCardinalBPARAM (Cardinal-package), 3
getCardinalNChunks (Cardinal-package), 3
getCardinalNumBlocks (deprecated), 7
getCardinalVerbose (Cardinal-package), 3
gridded (deprecated), 7
Hashmat (deprecated), 7
IAnnotatedDataFrame (deprecated), 7
IAnnotatedDataFrame-class (deprecated), 7
iData (deprecated), 7
iData, ANY-method (deprecated), 7
iData<-, (deprecated), 7
image, 26, 38
image (plot-image), 25
image, MSImagingExperiment-method (plot-image), 25
image, ResultsList-method (ResultsList-class), 37
image, SpatialCV-method (SpatialCV), 45
image, SpatialDGMM-method (SpatialDGMM), 46
image, SpatialFastmap-method (SpatialFastmap), 49
image, SpatialKMeans-method (SpatialKMeans), 51
INDEX

MSImageData-class (deprecated), 7
MSImageProcess (deprecated), 7
MSImageProcess-class (deprecated), 7
MSImageSet (deprecated), 7
MSImageSet-class (deprecated), 7
MSImagingArrays, 15, 16, 34, 66–69
MSImagingArrays
  (MSImagingArrays-class), 13
MSImagingArrays-class, 13
MSImagingExperiment, 13, 14, 34, 42, 68–70
MSImagingExperiment
  (MSImagingExperiment-class), 15
MSImagingExperiment-class, 15
mz,MassDataFrame-method
  (MassDataFrame-class), 10
mz,missing-method
  (MSImagingExperiment-class), 15
mz,MSImagingArrays-method
  (MSImagingArrays-class), 13
mz<-,MassDataFrame-method
  (MassDataFrame-class), 10
mz<-,MSImagingArrays-method
  (MSImagingArrays-class), 13
mzAlign (deprecated), 7
mzAlign,MSImagingExperiment,missing-method
  (deprecated), 7
mzAlign,MSImagingExperiment,numeric-method
  (deprecated), 7
mzBin (deprecated), 7
mzBin,MSImagingExperiment,missing-method
  (deprecated), 7
mzBin,MSImagingExperiment,numeric-method
  (deprecated), 7
mzData (deprecated), 7
mzData<-, (deprecated), 7
mzFilter (deprecated), 7
mzFilter,MSImagingExperiment-method
  (deprecated), 7
names,SpatialResults-method
  (SpatialResults-class), 58
names,SpectraArrays-method
  (SpectraArrays-class), 65
names,SpectralImagingArrays-method
  (SpectralImagingArrays-class), 66
names<-,SpectraArrays-method
  (SpectraArrays-class), 65
names<-,SpectralImagingArrays-method
  (SpectralImagingArrays-class), 66
names<-,SpectralImagingExperiment-method
  (SpectralImagingExperiment-class), 69
names<-,XDataFrame-method
  (XDataFrame-class), 76
NMF, 50, 55
NMF (SpatialNMF), 53
NMF,ANY-method (SpatialNMF), 53
NMF,SpectralImagingExperiment-method
  (SpatialNMF), 53
nnmf, 53
nnmf_als, 54
nnmf_mult, 54
normalization (deprecated), 7
normalization<-, (deprecated), 7
normalize, 17, 31, 35, 36, 44
normalize,SpectralImagingData-method
  (normalize), 17
nrun,PositionDataFrame-method
  (PositionDataFrame-class), 29
nrun,SpatialResults-method
  (SpatialResults-class), 58
nrun,SpectralImagingData-method
  (SpectralImagingData-class), 67
nscentroids, 62
OPLS, 46
OPLS (SpatialPLS), 56
opls, 58
OPLS,ANY-method (SpatialPLS), 56
OPLS,SpectralImagingExperiment-method
  (SpatialPLS), 56
parseAnalyze, 33, 34
parseImzML, 33, 34
PCA, 50, 54, 58
PCA (SpatialPCA), 54
PCA,ANY-method (SpatialPCA), 54
PCA,SpectralImagingExperiment-method
  (SpatialPCA), 54
pData, SpatialResults-method (SpatialResults-class), 58
pData, SpectralImagingData-method (SpectralImagingData-class), 67
pData <-, SpatialResults, ANY-method (SpatialResults-class), 58
pData <-, SpectralImagingData, ANY-method (SpectralImagingData-class), 67
peakAlign, 8, 18, 21–23, 33
peakAlign, MSImagingArrays-method (peakAlign), 18
peakAlign, MSImagingExperiment-method (peakAlign), 18
peakAlign, SpectralImagingArrays-method (peakAlign), 18
peakAlign, SpectralImagingExperiment-method (peakAlign), 18
peakBin (deprecated), 7
peakData (deprecated), 7
peakData <- (deprecated), 7
peakFilter (deprecated), 7
peakFilter, MSImagingExperiment-method (deprecated), 7
peakPick, 17, 19, 20, 22, 23, 31, 35, 36, 44
peakPick, MSImagingArrays-method (peakPick), 20
peakPick, MSImagingExperiment-method (peakPick), 20
peakPick, SpectralImagingData-method (peakPick), 20
peakPicking (deprecated), 7
peakPicking <- (deprecated), 7
peakProcess, 8, 19, 21, 22
peakProcess, MSImagingExperiment_OR_Arrays-method (peakProcess), 22
pixelApply (deprecated), 7
pixelApply, SpectralImagingExperiment-method (deprecated), 7
pixelData (SpectralImagingData-class), 67
pixelData, SpatialResults-method (SpatialResults-class), 58
pixelData, SpectralImagingData-method (SpectralImagingData-class), 67
pixelData <- (SpectralImagingData-class), 67
pixelData <-, SpatialResults-method (SpatialResults-class), 58
pixelData <-, SpectralImagingData-method (SpectralImagingData-class), 67
pixelNames (SpectralImagingData-class), 67
pixelNames (SpatialResults-class), 67
pixelNames, SpatialResults-method (SpatialResults-class), 58
pixelNames, SpectralImagingData-method (SpectralImagingData-class), 67
pixelNames <- (SpectralImagingData-class), 67
pixelNames <-, SpatialResults-method (SpatialResults-class), 58
pixelNames <-, SpectralImagingData-method (SpectralImagingData-class), 67
pixels, 24
pixels, SpectralImagingArrays-method (pixels), 24
pixels, SpectralImagingData-method (pixels), 24
pixels, SpectralImagingExperiment-method (pixels), 24
pixelSize (deprecated), 7
plot, 28
plot, plot-spectra, 27
plot, MeansTest, missing-method (MeansTest), 11
plot, MSImagingArrays, formula-method (plot-spectra), 27
plot, MSImagingArrays, missing-method (plot-spectra), 27
plot, MSImagingArrays, numeric-method (plot-spectra), 27
plot, MSImagingExperiment, character-method (plot-spectra), 27
plot, MSImagingExperiment, formula-method (plot-spectra), 27
plot, MSImagingExperiment, missing-method (plot-spectra), 27
plot, MSImagingExperiment, numeric-method (plot-spectra), 27
plot, ResultsList, ANY-method (ResultsList-class), 37
plot, ResultsList, missing-method (ResultsList-class), 37
plot, SpatialDGMM, missing-method (SpatialDGMM), 46
plot, SpatialFastmap, missing-method (SpatialFastmap), 49
reduceBaseline, SpectralImagingData-method
  (reduceBaseline), 35
rescale, 17
rescale_ref, 17
rescale_diff, 17
rescale_sum, 17
reset (process), 30
residuals, SpatialOPLS-method
  (SpatialPLS), 56
resolution (deprecated), 7
resolution<- (deprecated), 7
resultData (deprecated), 7
resultNames (deprecated), 7
ResultSet (deprecated), 7
ResultSet-class (deprecated), 7
ResultsList, 60
ResultsList (ResultsList-class), 37
ResultList-class, 37
risk.colors (deprecated), 7
rowMeans, SpectralImagingExperiment-method
  (summarizeFeatures), 73
rownames, SpectralImagingExperiment-method
  (SpectralImagingExperiment-class), 69
rownames<-, SpectralImagingExperiment-method
  (SpectralImagingExperiment-class), 69
rowStats, 74
rowStats, SpectralImagingExperiment-method
  (summarizeFeatures), 73
rowSums, SpectralImagingExperiment-method
  (summarizeFeatures), 73
run (PositionDataFrame-class), 29
run, PositionDataFrame-method
  (PositionDataFrame-class), 29
run, SpatialResults-method
  (SpatialResults-class), 58
run, SpectralImagingData-method
  (SpectralImagingData-class), 67
run<-, PositionDataFrame-method
  (PositionDataFrame-class), 29
run<-, PositionDataFrame-method
  (PositionDataFrame-class), 29
run<-, SpatialResults-method
  (SpatialResults-class), 58
run<-, SpectralImagingData-method
  (SpectralImagingData-class), 67
runNames (PositionDataFrame-class), 29
runNames, PositionDataFrame-method
  (PositionDataFrame-class), 29
runNames, SpatialResults-method
  (SpatialResults-class), 58
runNames, SpectralImagingData-method
  (SpectralImagingData-class), 67
runNames<-, PositionDataFrame-method
  (PositionDataFrame-class), 29
runNames<-, SpatialResults-method
  (SpatialResults-class), 58
runNames<-, SpectralImagingData-method
  (SpectralImagingData-class), 67
scanDirection (deprecated), 7
scanPattern (deprecated), 7
scanPolarity (deprecated), 7
scanType (deprecated), 7
segmentationTest (MeansTest), 11
selectROI, 26, 38
selectROI, SpectralImagingExperiment-method
  (selectROI), 38
setCardinalBPPARAM (Cardinal-package), 3
setCardinalNClims (Cardinal-package), 3
setCardinalNumBlocks (deprecated), 7
setCardinalVerbose (Cardinal-package), 3
sgmixn, 48
show, MSImagingArrays-method
  (MSImagingArrays-class), 13
show, MSImagingExperiment-method
  (MSImagingExperiment-class), 15
show, ResultsList-method
  (ResultsList-class), 37
show, SpatialResults-method
  (SpatialResults-class), 58
show, SpectraArrays-method
  (SpectraArrays-class), 65
show, SpectralImagingArrays-method
  (SpectralImagingArrays-class), 66
show, SpectralImagingData-method
  (SpectralImagingData-class), 67
show, SpectralImagingExperiment-method
  (SpectralImagingExperiment-class), 69
show, XDataFrame-method
  (XDataFrame-class), 76
SImageData (deprecated), 7
SIImageData-class (deprecated), 7
SIImageSet (deprecated), 7
SIImageSet-class (deprecated), 7
SimpleList, 37, 65
simspec, 42
simulateImage (simulateSpectra), 39
simulateSpectra, 39, 40
simulateSpectrum (deprecated), 7
slice, 42
smooth, 17, 31, 35, 36, 43
smooth, SpectralImagingData-method (smooth), 43
smoothing (deprecated), 7
smoothing<- (deprecated), 7
smoothSignal (deprecated), 7
smoothSignal, SpectralImagingExperiment-method (smooth), 43
spatial, 42
spatialApply (deprecated), 7
spatialApply, SpectralImagingExperiment-method (smooth), 43
SpatialCV, 45
SpatialCV-class (SpatialCV), 45
SpatialDGMM, 46
SpatialDGMM (SpatialDGMM), 46
SpatialDGMM, ANY-method (SpatialDGMM), 46
SpatialDGMM, SpectralImagingExperiment-method (SpatialDGMM), 46
SpatialDGMM-class (SpatialDGMM), 46
SpatialFastmap, 49
SpatialFastmap, 51, 54, 55
SpatialFastmap (SpatialFastmap), 49
SpatialFastmap, ANY-method (SpatialFastmap), 49
SpatialFastmap, SpectralImagingExperiment-method (SpatialFastmap), 49
SpatialFastmap-class (SpatialFastmap), 49
SpatialKMeans, 51
SpatialKMeans, 50, 52, 63
SpatialKMeans (SpatialKMeans), 51
SpatialKMeans, ANY-method (SpatialKMeans), 51
SpatialKMeans, SpectralImagingExperiment-method (SpatialKMeans), 51
SpatialKMeans-class (SpatialKMeans), 51
SpatialNMF, 53
SpatialNMF-class (SpatialNMF), 53
SpatialOPLS (SpatialOPLS), 56
SpatialOPLS-class (SpatialOPLS), 56
SpatialPCA, 54
SpatialPCA-class (SpatialPCA), 54
SpatialPLS, 56
SpatialPLS-class (SpatialPLS), 56
SpatialResults, 37
SpatialResults (SpatialResults-class), 37
SpatialResults-class, 58
SpatialShrunkenCentroids, 60
spatialShrunkenCentroids, 46, 52, 58
spatialShrunkenCentroids (SpatialShrunkenCentroids), 60
SpatialShrunkenCentroids, ANY, ANY-method (SpatialShrunkenCentroids), 60
SpatialShrunkenCentroids, ANY, missing-method (SpatialShrunkenCentroids), 60
SpatialShrunkenCentroids, SpectralImagingExperiment, ANY-method (SpatialShrunkenCentroids), 60
SpatialShrunkenCentroids, SpectralImagingExperiment, missing-method (SpatialShrunkenCentroids), 60
SpatialShrunkenCentroids-class (SpatialShrunkenCentroids), 60
spatialWeights, 10, 63
spatialWeights, ANY-method (spatialWeights), 63
spatialWeights, PositionDataFrame-method (spatialWeights), 63
spatialWeights, SpectralImagingExperiment-method (spatialWeights), 63
spectra, SpectralImagingData-method (SpectralImagingData-class), 67
spectra<-, SpectralImagingData-method (SpectralImagingData-class), 67
SpectraArrays, 14–16, 66–69
SpectraArrays (SpectraArrays-class), 65
SpectraArrays-class, 65
spectraData, SpectralImagingData-method (SpectralImagingData-class), 67
spectraData<-, SpectralImagingData-method (SpectralImagingData-class), 67
SpectralImagingArrays, 14, 68, 69
SpectralImagingArrays-class, 66
SpectralImagingArrays-class (SpectralImagingArrays), 66
<table>
<thead>
<tr>
<th>Function/Class</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpectralImagingData</td>
<td>14, 16, 65–67, 70</td>
</tr>
<tr>
<td>SpectralImagingData-class</td>
<td>67</td>
</tr>
<tr>
<td>SpectralImagingExperiment</td>
<td>16, 58, 68, 69</td>
</tr>
<tr>
<td>SpectralImagingExperiment-class</td>
<td>69</td>
</tr>
<tr>
<td>SpectraNames, SpectralImagingData-method</td>
<td>67</td>
</tr>
<tr>
<td>spectraNames&lt;-, SpectralImagingData-method</td>
<td>67</td>
</tr>
<tr>
<td>spectrapply, SpectralImagingData-method</td>
<td>70</td>
</tr>
<tr>
<td>spectrapply, MSIImagingArrays-method</td>
<td>70</td>
</tr>
<tr>
<td>spectrapply, MSIImagingExperiment-method</td>
<td>70</td>
</tr>
<tr>
<td>spectrapply, SpectralImagingArrays-method</td>
<td>70</td>
</tr>
<tr>
<td>spectrapply, SpectralImagingExperiment-method</td>
<td>70</td>
</tr>
<tr>
<td>spectraVariables, SpectralImagingData-method</td>
<td>67</td>
</tr>
<tr>
<td>spectrumRepresentation (deprecated)</td>
<td>7</td>
</tr>
<tr>
<td>spectrumRepresentation&lt;-, (deprecated)</td>
<td>7</td>
</tr>
<tr>
<td>subset, SpectralImagingArrays-method</td>
<td>72</td>
</tr>
<tr>
<td>subset, SpectralImagingExperiment-method</td>
<td>72</td>
</tr>
<tr>
<td>subsetFeatures, subsetFeatures</td>
<td>72</td>
</tr>
<tr>
<td>subsetPixels (subsetFeatures)</td>
<td>72</td>
</tr>
<tr>
<td>summarizeFeatures, summarizeFeatures,</td>
<td>8, 72, 73</td>
</tr>
<tr>
<td>summarizePixels</td>
<td>72</td>
</tr>
<tr>
<td>summarizePixels (summarizeFeatures)</td>
<td>73</td>
</tr>
<tr>
<td>svd</td>
<td>55</td>
</tr>
<tr>
<td>topFeatures (SpatialShrunkenCentroids)</td>
<td>60</td>
</tr>
<tr>
<td>topFeatures, MeansTest-method</td>
<td>(MeansTest), 11</td>
</tr>
<tr>
<td>topFeatures, ResultsList-method</td>
<td>(ResultsList-class), 37</td>
</tr>
<tr>
<td>topFeatures, SpatialKMeans-method</td>
<td>(SpatialKMeans), 51</td>
</tr>
<tr>
<td>topFeatures, SpatialOPLS-method</td>
<td>(SpatialOPLS), 56</td>
</tr>
<tr>
<td>topFeatures, SpatialPLS-method</td>
<td>(SpatialPLS), 56</td>
</tr>
<tr>
<td>topFeatures, SpatialShrunkenCentroids-method</td>
<td>(SpatialShrunkenCentroids), 60</td>
</tr>
<tr>
<td>vizi_engine (Cardinal-package)</td>
<td>3</td>
</tr>
<tr>
<td>vizi_par (Cardinal-package)</td>
<td>3</td>
</tr>
<tr>
<td>vizi_style (Cardinal-package)</td>
<td>3</td>
</tr>
<tr>
<td>vm_used, SpectraArrays-method</td>
<td>(SpectraArrays-class), 65</td>
</tr>
<tr>
<td>warp1</td>
<td>34</td>
</tr>
<tr>
<td>warp1_cow</td>
<td>35</td>
</tr>
<tr>
<td>warp1_dtw</td>
<td>35</td>
</tr>
<tr>
<td>warp1_loc</td>
<td>35</td>
</tr>
<tr>
<td>writeAnalyze, summarizationFeatures,</td>
<td>75</td>
</tr>
<tr>
<td>writeImzML, summarizeFeatures</td>
<td>75</td>
</tr>
<tr>
<td>writeImzML, MSImagingExperiment-method</td>
<td>(writeMSIData), 75</td>
</tr>
<tr>
<td>writeImzML, MSImagingExperiment.OR.Arrays-method</td>
<td>(writeMSIData), 75</td>
</tr>
<tr>
<td>writeMSIData</td>
<td>34, 75</td>
</tr>
<tr>
<td>XDataFrame, summarizeFeatures</td>
<td>11, 30</td>
</tr>
<tr>
<td>XDataFrame (XDataFrame-class)</td>
<td>76</td>
</tr>
<tr>
<td>XDataFrame-class</td>
<td>76</td>
</tr>
<tr>
<td>XDFrame (XDFrame-class)</td>
<td>76</td>
</tr>
<tr>
<td>XDFrame-class (XDFrame-class)</td>
<td>76</td>
</tr>
</tbody>
</table>

INDEX