Package 'uFTIR'

September 8, 2021

Type Package

Title Process and Analyze Agilent Cary 620 FTIR Microscope Images

Version 0.1.3

Date 2021-09-02

Maintainer Fabio Corradini <fabio.corradini@inia.cl>

Description A set of tools to read, process, and summarize Agilent Cary 620 uFTIR Microscope hyperspectral images
primarily intended for microplastic analysis.

License GPL-3

Encoding UTF-8

LazyData true

Depends R (>= 4.0.4.0), parallel (>= 4.0.4)

Imports Rcpp (>= 1.0.7), methods (>= 4.0.4), raster (>= 3.4.13), sp (>= 1.4.5)

Suggests testthat (>= 3.0.4), rgeos (>= 0.5.5), signal (>= 0.7.7), rgdal (>= 1.5.23)

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

Collate 'RcppExports.R' 'classes.R' 'clipper.R' 'datadocs.R' 'mask_sam.r' 'matrix_sam.r' 'mosaic_chunk.R' 'mosaic_compose.R' 'mosaic_info.R' 'mosaic_sam.R' 'plot_methods.R' 'preprocess.R' 'profile_methods.R' 'sam_load.R' 'sam_write.r' 'smooth_sam.R' 'summary_methods.R' 'tile_base_corr.R' 'tile_read.R' 'tile_sam.R' 'uFTIR.R' 'wavealign.R'

SystemRequirements C++11

NeedsCompilation yes

Author Fabio Corradini [aut, cre] (<https://orcid.org/0000-0001-9696-7643>)

Repository CRAN

Date/Publication 2021-09-08 13:30:02 UTC

R topics documented:

`as.clipper` *As Clipper*

Description

Coerce a `SAM` or `Smooth` object to clipper. A function useful to call `highlight_substance` when clipping is not necessary.

Usage

```
as.clipper(object, slice = 1)
```

Arguments

object	The SAM or Smooth object to be coerced.
slice	The slice to keep (a clipper is essentially a matrix).

Value

and S3 clipper

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
x <- tile_sam(x)
x <- as.clipper(x)
```

---

clipmask-class *Class holding info for the clipper*

Description

The filters we used to support our samples under the uFTIR microscope are round. This means that a cube might not be the best representation of our data, as they have noisy borders that might be missinterpreted by the program. Then, I implemented a clipper function to clip objects of class SAM or Smooth (even an S3 matrix) before summarizing them. Although this can be done step by step, you can also create a clip-mask for the summary_sam function to clip them for you. To crate this clip-mask, you can call the function toClip which returns an object of this class.

The class has yet another use. Since the samples are placed under the microscope by hand, the cropping area is not always the same and (usually) it has to be adjusted. To have a visual aid, you can use the function toClip, to then plot a SAM or Smooth and overlay the croppling circle by calling polygon and the xycoords slot of an object of this class (returned by toClip). Using that process you can test manually different center points and radius for the clipping circle.

Value

An S4 object of class "clipmask" to be used (internally) by the clipper function.

Slots

xycoords polygon (circle) coodinates to plot. matrix with two columns (x,y).

rad cicle radius. integer.

centre circle centre. Numeric vector of two elements (x,y).

See Also

toClip clipper

Examples

```
NULL
```

---

clipper *Clipper*

Description

A function to clip a SAM or Smooth object object (also a matrix, but I recommed this only for debugging). ALthough you can use this function in a 'step by step' process it is otherwise directly called by summary_sam which uses a clipmask object (returned by toClip as a clipping mask.

Usage

```
clipper(tarjet, centre = c(128, 128), rad = 120, slice = 1)
```

Arguments

tarjet	object of class SAM or Smooth (or matrix) to clip.
centre	The coordinates of the polygon centre (x,y).
rad	The circle radius.
slice	which slice of the SAM or Smooth object should be clipped? (only one!)

Value

The tarjet object clipped as matrix (and S3 clipper -not for human consumption).

See Also

toClip codeclipmask summary_sam.

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
x <- tile_sam(x)
x <- smooth_sam(x, as.integer(length(primpke@clusternames)), window = 3, 1)

clip <- toClip(8,20,c(10,10))
plot(x)
polygon(clip@xycoords)

x <- clipper(x, clip@centre, clip@rad, 1)
```

get_profile *Get profile*

Description

To check whether the program is doing a good job or not (an the library is appropriate), it is usually a good idea to recover the expectra of all the pixels that match a given cluster or substance to plot the spectra and compare it to a standard (the library entrance for that cluster/substance for example).

This function allows you to retrieve the spectra of all pixels that match a given cluster/substance and returns a matrix that can then be plotted to do the checking. It also allows the user to plot the object and overlay the points that match a feature. The latter can be achieved also by using the highlight_substance function. However, that function is only implemented for S3 "clipper" objects, so this might be a way to overcome that limitation.

get_profile_all will replace get_profile_sinfo some day. For now it does the same thing BUT it works only for clipper objects and you need to pass the fpa size and the wavenumber (waves) vector. It was designed to run in the cluster. It goes well with sam_write, sam_load.

Usage

```
get_profile(x, where, ...)

## S4 method for signature 'togetprofile,Tile'
get_profile(x, where, ...)

## S4 method for signature 'togetprofile,list'
get_profile(x, where, ...)

## S4 method for signature 'clipper,character'
get_profile(x, where, dst_cluster, fpa, waves)

get_profile_tile(
  x,
  where,
  dst_cluster,
  plotpol = TRUE,
  plotpt = FALSE,
  cluster = TRUE,
  slice = 1,
  clusternames = NULL,
  ...
)

get_profile_sinfo(
  x,
  where,
  dst_cluster,
```

```
  plotpol = TRUE,
  plotpt = FALSE,
  cluster = TRUE,
  slice = 1,
  clusternames = NULL,
  ...
)

get_profile_all(x, where, dst_cluster, fpa, waves)
```

Arguments

x	Object of class sam, Smooth or clipper that hold the processed tile/mosaic indicated in where.
where	Original object processed. It might by of class Tile or SpectralInfo. However, in the second case, where should be a labeled list of two elements one the SpectralInfo object (labeled "info"), and the other the tarjet dmd file (as a character labeled "dmdfile"). For get_profile_all it is a character vector with the *.dmd files that you want to retrieve.
...	other parameters for plot.raster.
dst_cluster	The expectra of the pixels that match which cluster/substance should be retrieved?
fpa	The fpa size, used in get_profile_all
waves	The wavenumbers, used in get_profile_all
plotpol	If true, the function will plot x and overlay countour lines highlighting dst_cluster. TRUE/FALSE.
plotpt	If true, the function will plot x and draw points over the pixels that match dst_cluster. For mosaics, the points will be only in the chunk selected in where. TRUE/FALSE.
cluster	If class(x) == "SAM", Should the substances or the clusters be used for analysis? TRUE means that the clusters are used, FALSE that the substances. If you set the argument to FALSE, dst_cluster and clusternames should be adjusted. Use the name of the substance and the substances list from the SpectralReference.
slice	If the object is of class sam or Smooth, Which slice of them should be considered when retrieving the pixel locations?
clusternames	if is.character(dst_cluster) you should provide a vector with the clusternames to coerse dst_cluster to numeric.

Value

matrix with the columns matching the wavenumbers of 'where'.

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
mosaic_sam(x, primpke, n_cores = 1)
```

```
y <- mosaic_compose(x@path, primpke@clusterlist)
y <- get_profile_sinfo(y,
                       where = list("info" = x,
                                    "dmdfile" = "mosaic_0000_0000.dmd"),
                       5, FALSE, FALSE)
```

`highlight_substance` *Highlight a selected substance*

Description

The function highlights a selected substance(s) -or cluster(s)- in a plot. It can add to an existing plot or create a new one. Currently implemented only for S3 objects returned by `clipper`.

Usage

```
highlight_substance(
  x,
  dst_cluster,
  new = TRUE,
  clusternames = NULL,
  polygon_color = "red",
  match_uFTIR = FALSE,
  ...
)
```

Arguments

`x`	"clipper" object to plot
`dst_cluster`	cluster (or substance) to highlight. Numeric and character inputs are supported. If is char, then you should provide clusternames to coerse it to numeric.
`new`	Do you want a new plot or the function should add the polygons to the current plot?
`clusternames`	char vector given the list of analyzed substance clusters. Only relevant when dst_cluster is a character.
`polygon_color`	color of the overploted polygons. Red by default.
`match_uFTIR`	The Agilent Microscope transposes and inverts the image (it inverts rows only - cols after transposition). Do you want the plot function to match this behaviour? Default FALSE.
`...`	other arguments for `plot_tile`

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
x <- tile_sam(x)
x <- smooth_sam(x, as.integer(length(primpke@clusternames)), window = 3, 1)
clip <- toClip(8,20,c(10,10))
plot(x)
polygon(clip@xycoords)
x <- clipper(x, clip@centre, clip@rad, 1)

highlight_substance(x, 5)
```

---

mask_sam *Mask SAM*

Description

The spectral angle mapper algorithm always find a match, as it was designed to compare between to spectra. However, somethimes the match is far from good and the user could want to drop the poorly matched pixels. This function will create a mask for such a purpose.

Usage

```
mask_sam(x, threshold = 1.1)
```

Arguments

x	an array. Tipically, the array at the raw_sam slot of a SAM object.
threshold	cut off value above wich, the SAM relation found is deemed unsignificant

Value

S3 matrix

Examples

```
set.seed(4356)
x <- array(abs(rnorm(1000)), c(10, 10, 10))
x <- mask_sam(x, 0.1)
# trick to plot as clipper
class(x) <- c("clipper", "matrix")
plot(x, legend = FALSE)
```

matrix_sam *Perform SAM between to matrices*

Description

Perform the spectral angle mapper algorithm between to matrices.

Usage

```
matrix_sam(x, y, x.wl, y.wl)
```

Arguments

x	matrix
y	matrix
x.wl	wavenumbers for x
y.wl	wavenumbers for y

Value

vector

mosaic_chunk *Mosaic chunk*

Description

The function read a single mosaic tile (.dmd file extension) using a SpectralInfo object as a guide to find the file. It returns an object of class Tile, which can be (pre/post)processed as if it were a single tile.

Usage

```
mosaic_chunk(info, path, dmdfile, fpa, wl)
```

Arguments

info	Optional. SpectralInfo object.
path	Optional. Path to dmdfile. Overwrites the path on the info object.
dmdfile	Target *.dmd file to read.
fpa	If info is missing, you should indicate the fpa size of the chunk.
wl	If info is missing, you should indicate the wavenumbers at which you read your data

Value

A Tile object.

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
y <- mosaic_chunk(info = x, dmdfile = "mosaic_0000_0000.dmd")
class(y)
```

---

mosaic_compose *Mosaic compose*

Description

There are to ways to process a mosaic. One way -which should be the standard- is to do it internally in R by calling first mosaic_info and later mosaic_sam.

The function will write in the path specified in the SpectralInfo object returned by mosaic_info a series of binary files holding the SAM results. In other words, the function does not load the results to R automaticaly. You can call this function to load them back to R in a single object, which will be of class SAM.

Usage

```
mosaic_compose(
  path = ".",
  clusterlist = NULL,
  nslices = NULL,
  drop_raw = FALSE
)
```

Arguments

path	Where are the binary files? you can use the 'path' slot of the SpectralInfo instead of enter it manually.
clusterlist	The clusterlist vector that you passed along with the SpectralReference in the call to mosaic_sam.
nslices	If you deal with large mosaics, you might want to load only a few of the sam matches. This argument allows you to define up to which match you want to load to R.
drop_raw	If you are not interested in the angles you can set this argument to TRUE and avoid load them to R.

Value

An object of class SAM.

See Also

mosaic_sam

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
mosaic_sam(x, primpke, n_cores = 1)
y <- mosaic_compose(x@path, primpke@clusterlist)
```

---

mosaic_info *Mosaic Info*

Description

The function is equivalent to tile_read. It makes an SpectralInfo object using the .dmt file. It does not load the measured spectra, and it was written for mosaic images only. See the page of SpectralInfo class for a full description of the returned object.

Usage

```
mosaic_info(dmtfile)
```

Arguments

dmtfile	Path to the .dmt file.

Value

An object of class SpectralInfo.

See Also

SpectralInfo

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
```

mosaic_sam *Mosaic SAM*

Description

The function performs the spectral angle mapper algorithm chunk by chunk for the mosaics files. It uses the parallel package, and by default all cores -1.

The function uses mosaic_chunk to cast a mosaic tile as a Tile object. Then performs a tile_base_corr, wavealing, and tile_sam to finaly write binary files that hold the Spectral Angle Mapper results in the destination folder (path slot of the SpectralInfo object). The files can be loaded back to R using link{mosaic_compose}.

The function is using the parallel package, as each tile_sam takes a while to complete.

Usage

```
mosaic_sam(
  info,
  sref,
  derivative = NULL,
  base_corr = TRUE,
  FUN = NULL,
  n_cores = NULL
)
```

Arguments

info	SpectralInfo object.
sref	SpectralReference object.
derivative	whether to apply the first (1) or second (2) derivative before sam. Default NULL.
base_corr	TRUE/FALSE should tile_base_corr be call before processing each chunk?
FUN	A function to be passed to preprocess.It is always applied as if 'data' were a SpectralPack object.
n_cores	The number of cores to parallelize the task. NULL means all cores -1.

Value

TRUE

See Also

For a single tile application see tile_sam.

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
mosaic_sam(x, primpke, n_cores = 1)
```

`plot,plot.uFTIR,missing-method` *Plotting Objects*

Description

Plotting method for objects of class `Tile`, `SpectralPack`, `SAM`, `Smooth`, and `clipper`.

In objects of class SAM, always the clusters slot gets plotted.

Usage

```
## S4 method for signature 'plot.uFTIR,missing'
plot(x, y, ...)

plot_tile(x, slice = 1, FUN = sum, match_uFTIR = FALSE, ...)
```

Arguments

`x`	Tile, SpectralPack, SAM, Smooth, or clipper to plot.
`y`	Missing.
`...`	Further arguments to `plot`
`slice`	For objects of class SAM or Smooth, Which slice of the cube should be plotted?
`FUN`	For objects of class SpectralPack and Tile, Which function should be used to collapse the cube to a matrix?
`match_uFTIR`	The Agilent Microscope transposes and inverts the image (it inverts rows only - cols after transposition). Do you want the plot function to match this behaviour? Default FALSE.

Details

The function requires the raster package to plot. It coerses the matrix to raster to do so.

Examples

```
# Tile objects:
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
plot(x)
# with arguments for raster::plot
plot(x, axes = FALSE, box = FALSE, legend = FALSE)

# SpectralPack objects:
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
plot(x)

# SAM objects:
```

```
x <- tile_sam(x)
plot(x)

# Smooth objects
x <- smooth_sam(x, as.integer(length(primpke@clusternames)), window = 3, 1)
plot(x)

# clipper objects:
clip <- toClip(8,20,c(10,10))
polygon(clip@xycoords)
x <- clipper(x, clip@centre, clip@rad, 1)
plot(x)
```

---

preprocess *Tile Preprocess*

Description

The function is a wrapper that allows the user to perform a user-define function to each pixel spectrum. It can be called internally in mosaic_sam using the FUN argument when processing mosaics.

As exemplified below, the function allows you to interact with other R packages that provide common features to analyze spectral data. Beware that you will need to adjust the wavenumbers manually if you are resampling... unless that 'data' is an object of class SpectralPack.

Usage

```
preprocess(data, FUN)

## S4 method for signature 'Tile,`function`'
preprocess(data, FUN)

## S4 method for signature 'SpectralPack,`function`'
preprocess(data, FUN)
```

Arguments

data	An object of class Tile or SpectralPack.
FUN	A function to apply to the Spectra slot.

Value

The same object but with the Spectra slot updated by FUN. SpectralPack objects return the wavelengths adjusted to the new extent, numbered from 1 to n (original values are lost).

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- preprocess(x, function(x){x+1})

# The function allows interacting with other R packages that provide common features
# for spectra analysis. For example, you can use the package "signal" to run
# a Savitzky-Golay filter.

if(requireNamespace("signal", quietly = TRUE)){

# for Tile objects. NOTE that after the preprocess x@wavenumbers does not match dim(x@Spectra)[3]
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- preprocess(x, function(x){signal::sgolayfilt(x)})
dim(x@Spectra)[3] == length(x@wavenumbers) # BEWARE!

# for SpectralPack objects
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
preprocess(x, function(x){signal::sgolayfilt(x)})

# Here the problem with the wavenumbers is gone
dim(x@Readings@Spectra)[3] == length(x@Readings@wavenumbers)
dim(x@Reference@Spectra)[3] == length(x@Reference@wavenumbers)
length(x@Readings@wavenumbers) == length(x@Reference@wavenumbers)

}
```

`primpke` *Example library*

Description

Example spectral library to be used in microplastic evaluations. An example on how to format a SpectralReference object.

It was taken form:

Primpke, S., Wirth, M., Lorenz, C., Gerdts, G. 2018. Reference database design for the automated analysis of microplastic samples based on Fourier transform infrared (FTIR) spectroscopy. Analytical and Bioanalytical Chemistry 410: 5131-5141. You might access to the article at doi: 10.1007/s00216-018-1156-x

Usage

```
primpke
```

Format

an S4 SpectralReference object with:

substances character vector with the 270 names of the polymers included in Spectra

clusterlist integer vector with the corresponding cluster of each polymer described

clusternames the name of the cluster, sorted in ascending order according to clusterlist

Spectra matrix with the absorbance readings of each polymer (270 x 1176)

wavenumbers read wavenumbers. Correspond to Spectra ncol

Source

doi: 10.1007/s00216-018-1156-x

SAM-class *Sam Prediction*

Description

The class holds the resuls of the Spectral Angle Mapping algorithm and it originates by a call to tile_sam or to mosaic_compose. It has three slots with congruent cubes. Each cube has its first two dimensions equal to the FPA size of the uFTIR Microscope image processed and the third dimension equal to the number of substances in the SpectralReference passed when calling tile_sam or mosaic_sam. This class have methods to plot, summarize, and extract the readings of pixel matching a given SpectralReference substance. You can use the generics plot, summary, and get_profile but beware, you might have to provide extra arguments! (so, please don't see only the generic but the method for this class, see the section see also)

The sam object can be postprocessed by calling smooth_sam and/or clipped to a given extent by calling clipper (currently only circular extents are supported).

Value

An S4 object of class SAM

Slots

raw_sam a cube -array-. The numeric score for each SpectralReference substance passed by the call (slices), for each pixel of the FPA array of the Tile (rows and cols).

substances a cube -array-. The slices of the cube hold the substances (as number) of the SpectralReference object passed when calling tile_sam or mosaic_sam. The substance number are sorted in a decreacing order. In this way, the first slide is the best match, the second the second-best, and so on according to the Spectral Angle Mapping algorithm.

clusters a cube -array-. Equivalent to the cube in the substances slot but instead of a substance number it has per ech slice the cluster number that corresponds to the clusterlist slot of the SpectralReference object.

See Also

plot_tile summary_sam get_profile_tile get_profile_sinfo

Examples

```
NULL
```

`sam_load` *Sam load*

Description

The inverse function for sam_write

Usage

```
sam_load(dmdfile)
```

Arguments

dmdfile	a connection to the .bin or .ban file (yes, is true).

Value

An object of SAM

`sam_write` *Write a sam object to disk*

Description

A wrapper to the C++ mosaic_sam_write

Usage

```
sam_write(tile, sam)
```

Arguments

tile	The tile form which the sam originates
sam	The sam that you want to write to disk

Smooth-class *Smooth*

Description

An object of this class is returned by a call to smooth_sam. It holds a smoother version of a SAM object, keeping only the clusters slot and slicing it to a user-given number of slices. This class have methods to plot, summarize, and extract the readings of pixel matching a given SpectralReference substance. You can use the generics plot, summary, and get_profile but beware, you might have to provide extra arguments! (so, please don't see only the generic but the method for this class, see the section see also).

Value

An S4 object of class "Smooth.

Slots

smooth a cube -array- (even if only has 1 slice) that holds a smooth version of a SAM clusters slot.

See Also

plot_tile summary_sam get_profile_tile get_profile_sinfo

Examples

```
NULL
```

smooth_sam *Smooth a Spectral Angle Mapper output*

Description

The function makes a smooth version of the object returned by tile_sam or mosaic_compose. The function smooths out only the cluster slot, slicing it to a user-given range of slices.

Usage

```
smooth_sam(x, nclusters, window = 5, nslices = 1)
```

Arguments

x	Object of class SAM.
nclusters	How many clusters did the spectral library had? This parameter refer back to the length of the clusterlist slot of the SpectralReference object used when calling tile_sam or mosaic_sam. It should be an integer.
window	The function smooths out the SAM objectc "x" using a moving window. You should decide which size (in pixels) this window should be.
nslices	Integer. Starting from 1, up to which slice of the cube holded in the clusters slot of the SAM objectc "x" do you want the smooth_sam function to work? It will influence the output. See return.

Value

An S4 object of class Smooth. It has only one slot (smooth) that holds an array in which each slice is the smooth version of the corresponding slice of a SAM object.

The returned object has methods to plot, summarize, and extract profiles by calling the generics plot, summary, and get_profile. You might have to provide extra arguments. You can refer to the see also section to look for the object specific methods.

See Also

plot_tile summary_sam get_profile_tile get_profile_sinfo

Examples

```
a <- matrix(c(1,1,1,1,3,3,
              2,2,2,1,2,3,
              3,1,1,2,3,1,
              3,3,2,2,1,1,
              1,1,1,3,3,2,
              3,3,3,2,2,3), nrow = 6, byrow = TRUE)
a <- array(c(a,a), dim = c(6,6,2))
test_sam <- new(Class = "SAM",
                raw_sam = a,
                substances = a,
                clusters = a)

smooth_sam(test_sam, nclusters = 3, window = 3)
```

SpectralInfo-class *Spectral info*

Description

General class that holds basic information about an image. Usually used when working with mosaics, as in this case the image is not read/loaded to R until is processed (e.g. by running a mosaic_sam process). Currently, the only function that returns an object of this class is he `mosaic_info` function. The object can by used later to read/load to R a single mosaic chuck using the `mosaic_chunk` function, or to process the whole mosaic image by calling the `mosaic_sam` function.

The class `Tile` contains the "Spectral_info" class. By this means, an object of class `Tile` holds the same information about the reading, and an extra slot called "Spectra" that holds the actual readings.

Value

An S4 object of class "Spectral_info"

Slots

`file` character. The full path to the .dmt or .bsp file.

`date` POSIXct. The date in which the image (mosaic or tile) was taken.

`fpasize` numeric. The size of the focal plane array used when measuring.

`wavenumbers` numeric. The wavenumbers at which the tarject image (mosaic or tile) was taken.

`path` character. The path to the folder that holds all the uFTIR Microscope output files.

See Also

`Tile`

Examples

```
NULL
```

SpectralPack-class *Spectral Pack*

Description

A call to the function `wavealign` (specifically a call to `TileRead.wavealign`) returns an object of class `SpectralPack`. This class reunites the uFTIR Microscope readings loaded to R by a call to `tile_read` with a tarjet spectral library of class `SpectralReference`. Since this object is an output of a call to `wavealign` both the `Tile` object and `SpectralReference` have congruent wavenumbers. See `wavealign` how the resampling is done when calling different methods. A `SpectralPack` object is used as imput when calling `tile_sam` to run the Spectral Angle Mapping algorithm.

Value

An S4 object of class "SpectralPack" ready for spectral angle mapping.

Slots

`Readings` an object of class `Tile`.

`Reference` an object of class `SpectralReference`.

Examples

```
NULL
```

`SpectralReference-class`
Spectral Reference Class

Description

I made this class to hold spectral libraries. This libraries can be used to match a know spectrum with a pixel reading. `SpectralReference` objects are used by several functions in the package, at all analysis stages (i.e. preprocessing, processing, and summarizing/plotting). You can see an example in the data `primpke` (you can load it by calling data("primpke")). The format includes character/integer vectors to hold the names of the polymers, cluster IDs, and cluster names, all of which should correspond to one another. It also hold a matrix with the spectra of each polymer, with rows matching the vectors provided and a fourth vector holding the wavelengts (that should be equal to the matrix cols).

The example provided in the package was taken form Primpke, S., Wirth, M., Lorenz, C., Gerdts, G. 2018. Reference database design for the automated analysis of microplastic samples based on Fourier transform infrared (FTIR) spectroscopy. Analytical and Bioanalytical Chemistry 410: 5131-5141. You might access to the article at doi: 10.1007/s00216-018-1156-x.

If you have reference data that was taken considering a different range of wavenumbers, you should resample it first for the wavenumbers to match.

Value

A S4 object of class "SpectralReference".

Slots

`substances` A character vector identifying by name the polymers whose spectrum is provided in Spectra. The order of the elements should be consistent with clusterlist, clusternames and Spectral rows. Along the same lines, the object expects length(substances) == nrow(Spectra).

`clusterlist` If the polymers are aggregated in clusters this slot should hold an integer vector with the correspondent cluster for each polymer individuated in substances. If you don't want to use clusters or you don't have clusters for your polymers/library you can place a sequence of integers from 1:nrow(Spectra) (e.g. by calling seq_along(substances)) in this slot.

`clusternames` Character vector holding reference names for each cluster, sorted according to clusterlist. In other words, if you want to name cluster 1L "polystyrene" the first position in clusternames should be equal to "polystyrene". If you don't want to use clusters or you don't have clusters for your polymers/library you can duplicate substances. The program expects that the length of the clusternames vector equals the length of unique elements in clusterlist.

`Spectra` A matrix holding row-wise the spectrum of each substance. Each row should correspond to the spectrum of one substance. In the same lines, columns should hold recorded measures for each wavenumber. The program expects that the length of the substance vector equals the number of rows of Spectra and that the number of colums of Spectra equals the length of the wavenumbers vector.

`wavenumbers` A numeric vector with the wavenumbers.

Examples

```
data("primpke")
```

`summary,summary_sam-method`

Method to summarize `SAM` *or* `Smooth` *objects.*

Description

Summary method for SAM and Smooth objects.

Usage

```
## S4 method for signature 'summary_sam'
summary(object, ...)

summary_sam(
  object,
  mask = NULL,
  clusternames = NULL,
  slice = 1,
  window = NULL,
  smooth = TRUE,
  temporal = FALSE
)
```

Arguments

`object`	An object of class `SAM` or `Smooth`. It accepts also object of S3 class "clipper" returned by the `clipper` function.
`...`	other parameters for summary_sam
`mask`	a clipper mask returned by `toClip` function. It should be set to NULL To avoid clipping, the default.
`clusternames`	The name of the clusters, it should match the clusternames list comprised in the `SpectralReference` object passed to `tile_sam` or `mosaic_sam`. Mandatory if object is of class `SAM` when smooth = TRUE.
`slice`	Which slice of object should be summarized?
`window`	When smooth = TRUE the size of the window for `smooth_sam`.

smooth	When object if of class SAM, should the function silently apply smooth_sam before summarizing?.
temporal	if TRUE the raster and shape files from which the summary is extracted are written in the working directory.

Value

A dataframe with three columns showing the cluster number, the area in pixels^2, and the cluster name as character (only if the parameeter clusternames is within the function call). Each row stand for a single particle. The function writes on disk two files "raster_out.tif" which is a raster of the object passed through the object paramenter, and a shape_out "ESRI Shapefile" folder with a shapefile holding a vectorized version of "raster_out.tif".

Examples

```
x <- mosaic_info(base::system.file("extdata/mosaic.dmt", package = "uFTIR"))
mosaic_sam(x, primpke, n_cores = 1)
y <- mosaic_compose(x@path, clusterlist = primpke@clusterlist)
summary_sam(y, clusternames = primpke@clusternames, smooth = FALSE, temporal = TRUE)
```

Tile-class *Tile*

Description

Class to hold the reading of a single tile. It can also store a single tile of a mosaic, to which we will refer to as chunks. It is an extended class for SpectralInfo, adding an extra slot to hold the uFTIR Microscope readings -called Spectra. The functions tile_read and mosaic_chunk return an object of this class. The class has a defined method to plot, which can be called either by the generic plot or by plot_tile. You can use an object of this class in the following functions:

plot_tile To plot a heat map of the reading. Available too through the generic plot.

tile_base_corr A preprocessing function that replace negative values with zeros.

wavealign A resampling method that allows to match the wavenumbers of a given tile/chunk with the wavenumbers of a SpectralReference.

tile_sam When a Tile object is linked to a SpectralReference by a former call to wavealign, this function performs a spectral angle mapper match between the Tile and the SpectralReference.

get_profile_tile To get the pixels spectra that matched a given polymer/cluster.

Value

An S4 object of class "Tile".

Slots

Spectra array. The aquired spectra (.dat or dmd file)

Examples

```
NULL
```

`tile_base_corr` *Baseline correction*

Description

A function to substract the minimun of a pixel (vector) to have a minimal signal = 0, avoiding negative readings. Currently the only preprocessing method implemented.

Usage

```
tile_base_corr(data)
```

Arguments

`data`	Object of class `Tile`.

Value

An object of class `Tile` congruent with data, with the spectra modified.

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
```

`tile_read` *Single tile read*

Description

The function loads a single tile to R as an object of class `Tile`.

The function is also capable to load the toy mosaic produced by the Agilent sofware. You can do so by passing as bspfile parameter the .bsp file comprised in a mosaic folder. However, beware! the toy mosaic IS NOT the complete mosaic, it is only a reduced version of it (i.e. you will lose resolution, and you will not longer know the pixel size). Use it with care.

Usage

```
tile_read(bspfile)
```

Arguments

`bspfile`	path to the .bsp file.

Value

An S4 Object of class Tile.

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
```

`tile_sam` *Spectral Angle Mapper*

Description

Performs the Spectral Angle Mapper to match the SpectralReference with yout readings (as Tile).

Usage

```
tile_sam(SpectralPack, derivative = NULL)
```

Arguments

SpectralPack	an object of class SpectralPack.
derivative	whether to apply the first (1) or second (2) derivative to preprocess the data before the algorithm is applied. Default NULL (It tells the program to not derivate).

Value

An object of class SAM.

See Also

For its application to mosaic images see mosaic_sam.

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
x <- tile_sam(x)
```

toClip *Visual clipper*

Description

Visual aid to find the right center and radius for the clipper function. It does not clip, but return an usable (plotable) object.

Since the samples are placed under the uFTIR Microscope by hand, the cropping area to pass to clipper is not always the same and (usually) it has to be adjusted. To have a visual aid, you can use this function. It returns an S4 object of class clipmask which can be over-plotted on top of a SAM or Smooth object by calling polygon and using the xycoords slot of the returned object. Using that process you can test manually different center points and radius for the clipping circle.

Usage

```
toClip(rad = 1, segments = 5, centre = c(0, 0))
```

Arguments

rad	The circle radius.
segments	How many segments should the resulting polygon have? (because it is not a circle, 20 is ok).
centre	The coordinates of the polygon centre (x,y).

Value

An object of class clipmask.

See Also

clipper

Examples

```
toClip(1, 5, c(0,0))

x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
x <- tile_sam(x)
x <- smooth_sam(x, as.integer(length(primpke@clusternames)), window = 3, 1)

clip <- toClip(8,20,c(10,10))
polygon(clip@xycoords)

x <- clipper(x, clip@centre, clip@rad, 1)
```

uFTIR *Process and analyze Agilent Cary 620 FTIR Microscope images*

Description

Process and analyze Agilent Cary 620 FTIR Microscope images

wavealign *Methods to resample the wavenumbers*

Description

It clips and resamples the wavenumbers of data.x and data.y to a common extent. Typically, it uses the wavenumbers of data.x to resample data.y and then the function clips the largest base on the narrowest.

Usage

```
wavealign(data.x, data.y)

## S4 method for signature 'numeric,SpectralReference'
wavealign(data.x, data.y)

## S4 method for signature 'Tile,SpectralReference'
wavealign(data.x, data.y)

## S4 method for signature 'SpectralReference,Tile'
wavealign(data.x, data.y)

MosaicGetRef.wavealign(data.x, data.y)

TileRead.wavealign(data.x, data.y)
```

Arguments

data.x	An object of class SpectralReference or Tile.
data.y	An object of class SpectralReference or Tile. It should be the other one.

Details

There are two methods defined:

- **TileRead.wavealign** - This method should be the only one used by the user.
- **MosaicGetRef.wavealign** - This method is called internally by mosaic_sam. The method is different since it expect a numeric vector in data.x, which is taken from the wavenumbers slot of a SpectralInfo object. The implication is that, for mosaics, the SpectralReference is always resampled according to the wavenumbers of the mosaics. It was programed this way, since mosaics are not loaded into the memory (R) until they are processed by mosaic_sam.

Value

- **TileRead.wavealign** - An object of class SpectralPack
- **MosaicGetRef.wavealign** - The SpectralReference object passed as data.y, resampled to data.x.

Examples

```
x <- tile_read(base::system.file("extdata/tile.bsp", package = "uFTIR"))
x <- tile_base_corr(x)
x <- wavealign(x, primpke)
```

Index