Package 'EEM'

November 5, 2015

Type Package

Title Read and Preprocess Fluorescence Excitation-Emission Matrix (EEM) Data

Version 1.0.4

Date 2015-10-14

Author Vipavee Trivittayasil

Maintainer Vipavee Trivittayasil <vipavee.tri@gmail.com>

Description Read raw EEM data and prepares them for further analysis.

Depends R (>= 3.0.0)

Imports tools, reshape2, graphics, colorRamps, utils, readxl, R.utils, sp

Suggests stats, pls, knitr, testthat

VignetteBuilder knitr

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2015-11-05 08:31:17

R topics documented:

`applejuice` *Apple juice*

Description

Apples of each of six types (Aomori–Fuji, Aomori–Jona, Aomori–Ohrin, NZ–Envy, NZ–Jazz, NZ–Fuji) were blended and filtered using a gauze. Fluorescence profiles of complete excitation–emission matrix of filtered solutions (diluted with water to 147 times) were measured using fluorescence spectroscopy machines. The sample name refers to "type–fruit number–replicate". To save space, only two apples of each types were given in the dataset.

Usage

```
data("applejuice")
```

Examples

```
data(applejuice)
summary(applejuice)
```

`cutEEM` *Cut portions of EEM*

Description

Cut portions of EEM

Usage

```
cutEEM(x, cutEX = NULL, cutEM = NULL)

## S3 method for class 'EEM'
cutEEM(x, cutEX = NULL, cutEM = NULL)

## S3 method for class 'EEMweight'
cutEEM(x, cutEX = NULL, cutEM = NULL)
```

Arguments

x	a list of EEM data generated by readEEM function or EEMweight object generated by extract-related functions.
cutEX	Numeric or sequential data specifying regions to be cut for excitation wavelength. Examples, 200 or 200:500 or c(200:300, 600:800)
cutEM	Numeric or sequential data specifying regions to be cut for emission wavelength. Examples, 200 or 200:500 or c(200:300, 600:800)

Value

A list similar to input EEM is returned but with specified portions cut.

Examples

```
data(applejuice)
drawEEM(cutEEM(applejuice, cutEX = 200:250), 1)
```

---

delScattering *Delete scattering rays*

Description

This function deletes two types of rays that are not related to fluorescence emission: (1) regions where emission wavelength is shorten than excitation light, (2) scattering rays and their second, third and forth order lights.

Usage

```
delScattering(EEM, rep = 0, first = 30, second = 40, third = 40,
  forth = 0)
```

Arguments

EEM	A list containing EEM data as created by readEEM function.
rep	(optional) Regions to be deleted are to be replaced with rep: 0 or NA
first	(optional) Width of region to be deleted for first order scattering rays [nm]
second	(optional) Width of region to be deleted for second order scattering rays [nm]
third	(optional) Width of region to be deleted for third order scattering rays [nm]
forth	(optional) Width of region to be deleted for forth order scattering rays [nm]

Value

A list similar to input EEM is returned but with all scattering rays deleted.

References

Fujita, K., Tsuta, M., Kokawa, M., and Sugiyama, J. (2010). Detection of deoxynivalenol using fluorescence excitation–emission matrix. Food and Bioprocess Technology, 3(6), 922–927.

Examples

```
data(applejuice)
drawEEM(delScattering(applejuice, NA), 1)
```

---

drawEEM *Draw contour for EEM data*

Description

This function is a wrapper function for `filled.contour` to draw contour for EEM data.

Usage

```
drawEEM(x, ...)

## S3 method for class 'EEM'
drawEEM(x, n, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", color.palette = matlab.like,
  nlevels = 50, main = NULL, flipaxis = FALSE, ...)

## S3 method for class 'EEMweight'
drawEEM(x, ncomp, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", color.palette = matlab.like,
  nlevels = 50, main = NULL, flipaxis = FALSE, ...)

## S3 method for class 'matrix'
drawEEM(x, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", color.palette = matlab.like,
  nlevels = 50, main = NULL, flipaxis = FALSE, ...)

## S3 method for class 'data.frame'
drawEEM(x, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", color.palette = matlab.like,
  nlevels = 50, main = NULL, flipaxis = FALSE, ...)

## S3 method for class 'numeric'
drawEEM(x, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", color.palette = matlab.like,
  nlevels = 50, main = NULL, flipaxis = FALSE, ...)
```

Arguments

x	a list of EEM data generated by readEEM function or EEMweight object generated by extract-related functions.
...	(optional) further arguments passed to other methods of filled.contour
n	sample number. The number should not exceed length(EEM)
exlab	(optional) excitation-axis label
emlab	(optional) emission-axis label
color.palette	(optional) contour color palette. See palette for more details
nlevels	(optional) number of levels used to separate range of intensity value
main	(optional) plot title
flipaxis	(optional) flip axis
ncomp	number of components

Value

A figure is returned on the graphic device

Methods (by class)

- EEM: draw contour of EEM data created by readEEM function
- EEMweight: draw contours of the output from getLoading and getReg.
- matrix: draw contour of a matrix with columns being excitation wavelength and rows being emission wavelength
- data.frame: draw contour of a data frame with columns being excitation wavelength and rows being emission wavelength
- numeric: draw contour of a vector of numeric values which have names in the format of EX...EM...

Examples

```
# method for class "EEM"
data(applejuice)
drawEEM(applejuice, 1) # draw contour of the first sample
drawEEM(applejuice, 1, flipaxis = TRUE) # flip the axis

# method for class "EEMweight"
applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
drawEEM(getLoading(result), 1) # plot loading of the first PC
```

EEM *EEM: A package for reading and preprocessing fluorescence excitation-emission matrix*

Description

EEM package can be used to import raw data files, visualizing data and preparing them for multivariate analysis

EEM-misc *Internal functions for EEM package*

Description

Internal functions for EEM package

Usage

```
generatePoint(n, pch = NULL)

generateColor(n, color.palette = NULL)

getEX(string, digits = NULL)

getEM(string, digits = NULL)
```

Arguments

n	number
pch	Either an integer specifying a symbol or a single character to be used as the default in plotting points.
color.palette	(optional) contour color palette. See palette for more details
string	string or vector of strings
digits	integer indicating the number of decimal places (round) or significant digits (signif) to be used. Negative values are allowed (see 'Details').

Details

'generatePoint' and 'generateColor' are used to create point and color vector from specified number (n) and palette.

Functions

- generateColor:
- getEX:
- getEM:

extract *Extract values from other models*

Description

Extract values from other models

Usage

```
getLoading(x)

getReg(x)
```

Arguments

x	output variable from prcomp or plsr functions

Value

A 'EEMweight' list containing title and value attributes.

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
loading <- getLoading(result)
str(loading)
```

findLocalMax *Find local maximum peaks*

Description

Find local maximum peaks in EEM data

Usage

```
findLocalMax(data, ...)

## S3 method for class 'EEM'
findLocalMax(data, n, threshold = 0.7, ...)

## S3 method for class 'matrix'
findLocalMax(data, threshold = 0.7, ...)
```

```
## S3 method for class 'data.frame'
findLocalMax(data, threshold = 0.7, ...)

## S3 method for class 'numeric'
findLocalMax(data, threshold = 0.7, ...)
```

Arguments

data	EEM data generated by readEEM function, or a matrix or dataframe with columns = excitation and rows = emission wavelengths.
...	(optional) further arguments passed to other methods
n	sample number. The number should not exceed length(EEM).
threshold	threshold value ranging from 0 ~ 1. Lower the value to cover low peaks.

Value

Print a dataframe of local maximum peaks and return a character vector of peak names.

Methods (by class)

- EEM: for EEM data created by readEEM function
- matrix: for a matrix with columns being excitation wavelength and rows being emission wavelength
- data.frame: for a dataframe with columns being excitation wavelength and rows being emission wavelength
- numeric: for a vector of numeric values which have names in the format of EX...EM...

Examples

```
data(applejuice)
findLocalMax(applejuice, 1)
```

---

fold *Fold EEM matrix into a list*

Description

Fold EEM matrix into a list

Usage

```
fold(EEM_uf)

## S3 method for class 'matrix'
fold(EEM_uf)

## S3 method for class 'numeric'
fold(EEM_uf)
```

Arguments

EEM_uf	Unfolded EEM matrix where columns are wavelength condition and rows are samples. It should have corresponding column names (formatted as EX###EM###) and row names.

Value

EEM a list containing EEM/EEM data

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
applejuice_uf_norm <- normalize(applejuice_uf) # normalize matrix
drawEEM(fold(applejuice_uf_norm), 1) # visualize normalized EEM
```

---

gluten *Gluten*

Description

Pure wheat gluten and pure wheat starch were mixed at gluten ratios ranging from 0 to 100 %, in 20 % increments. The samples were set in a cell with a quartz glass window, and the samples were pressed against the glass to obtain a flat surface. This dataset contains fluorescence excitation-emission profiles of each samples with 8 replicates. To save space, only the data with gluten ratios ranging from 0 to 60 % was provided.

Usage

```
data("gluten")
```

References

Kokawa, M., Fujita, K., Sugiyama, J., Tsuta, M., Shibata, M., Araki, T., & Nabetani, H. (2012). Quantification of the distributions of gluten, starch and air bubbles in dough at different mixing stages by fluorescence fingerprint imaging. Journal of Cereal Science, 55(1), 15–21.

Examples

```
data(gluten)
summary(gluten)
```

normalize *Normalize data*

Description

Normalize data (area under the curve = 1)

Usage

```
normalize(EEM_uf)
```

Arguments

EEM_uf	Unfolded EEM matrix where columns are wavelength condition and rows are samples

Details

The unfolded EEM data can be normalized by dividing each variable by the sum of the absolute value of all variables in a sample, such that the summation of absolute values of all variables in each sample was equal to 1. This is can be used to reduce the scaling difference,which is common in spectroscopic applications. This difference is usually caused by the scattering effect, source/detector variation and instrumental sensitivity.

Value

A matrix of normalized data

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
applejuice_uf_norm <- normalize(applejuice_uf) # normalize data

rowSums(abs(applejuice_uf_norm), na.rm = TRUE) # the absolute sum of each row equal to 1
```

plotLoading *Plot loadings for EEM data*

Description

Plot loadings for EEM data

Usage

```
plotLoading(x, ncomp = NULL, ...)
```

Arguments

x	output variable from prcomp or plsr functions
ncomp	number of components
...	(optional) arguments for drawEEM and filled.contour

Value

A figure is returned on the graphic device

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
plotLoading(result, ncomp = 1) # plot loading of the first PC
```

---

plotReg *Plot regression coefficients for EEM data*

Description

Plot regression coefficients for EEM data

Usage

```
plotReg(x, ncomp = NULL, ...)
```

Arguments

x	output variable from plsr function
ncomp	number of components
...	(optional) arguments for drawEEM and filled.contour

Value

A figure is returned on the graphic device

Examples

```
data(gluten)
gluten_uf <- unfold(gluten) # unfold list into matrix

# delete columns with NA values
index <- colSums(is.na(gluten_uf)) == 0
gluten_uf <- gluten_uf[, index]
gluten_ratio <- as.numeric(names(gluten))
```

```
require(pls)
model <- plsr(gluten_ratio ~ gluten_uf, ncomp = 3)
plotReg(model)
```

---

plotScore *Plot score for prcomp result*

Description

Plot score for prcomp (PCA) result

Usage

```
plotScore(prcompResult, xPC = 1, yPC = 2, group = NULL, cex = 1.5,
  label = NULL, pos = 4, col = NULL, pch = NULL,
  legendlocation = "bottomright", legendoutside = FALSE,
  rightwhitespace = 0, legendinset = 0, ...)
```

Arguments

prcompResult	output object from prcomp function
xPC	an integer indicating PC component on x-axis
yPC	an integer indicating PC component on y-axis
group	variable of numeric, character or factor class separating the samples into groups. Able to accept up to two groups which can be stated by group = c(group1, group2)
cex	(optional) size of points on graphs
label	(optional) a character vector or expression specifying the text to be written.
pos	(optional, applicable when label is given) a position specifier for the text. If specified this overrides any adj value given. Values of 1, 2, 3 and 4, respectively indicate positions below, to the left of, above and to the right of the specified coordinates.
col	point color
pch	point type
legendlocation	(optional)location of legend on graph. Look up legend for more details.
legendoutside	(optional) set to TRUE if you want to put legend on the outside of the plot. The legend location is defaulted to topright.
rightwhitespace	(optional) set width for white space for legend. Only applicable if legendoutside = TRUE
legendinset	(optional) how much legend box will be pushed to the right. Only applicable if legendoutside = TRUE
...	additional arguments for par

Value

A figure is returned on the graphic device

See Also

plotScorem

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
plotScore(result, xPC = 1, yPC = 2) # plot PC1 vs PC2 score
plotScore(result, xPC = 1, yPC = 2, pch = 3, col = "blue") # change shape and color

# get country of apple production
country <- sapply(strsplit(names(applejuice), split = "-"), "[", 1)
plotScore(result, xPC = 1, yPC = 2, label = country) # add label

# or plot by group
plotScore(result, xPC = 1, yPC = 3, group = country)

# custom point types and color
plotScore(result, xPC = 1, yPC = 3, group = country, pch = c(1,2), col = c("green", "black"))

# move legend outside
plotScore(result, xPC = 1, yPC = 3, group = country, legendoutside = TRUE)
```

---

plotScorem *Plot score matrix for prcomp result based on group*

Description

Plot score matrix for prcomp (PCA) result based on group

Usage

```
plotScorem(prcompResult, ncomp = 4, group, cex = 1.5, col = NULL,
  pch = NULL, legendtitle = NULL, ...)
```

Arguments

prcompResult	output object from prcomp function
ncomp	maximum number of PC score to plot
group	variable of numeric, character or factor class separating the samples into groups.
cex	(optional) size of points on graphs
col	point color

pch	point type
legendtitle	legend title
...	additional arguments to be passed on to pairs

Value

A figure is returned on the graphic device

See Also

pairs, plotScore

Examples

```
data(applejuice)
# country of apple production
country <- sapply(strsplit(names(applejuice), split = "-"), "[", 1)

applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
# plot PC1 vs PC3 score based on country of production
plotScorem(result, ncomp = 4, group = country)

# specify colours
plotScorem(result, ncomp = 4, group = country, col = c("black", "grey"))
```

---

prcompname *Create name for prcomp result*

Description

Create name for prcomp result

Usage

```
prcompname(prcompResult, PC, explvar = TRUE)
```

Arguments

prcompResult	output value from prcomp function
PC	PC number
explvar	(logical) show explained variance (%) or not

Value

String

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
result <- prcomp(applejuice_uf)
prcompname(result, 1)
```

`print.EEM` *Print EEM*

Description

Print EEM

Usage

```
## S3 method for class 'EEM'
print(x, ...)
```

Arguments

`x`	EEM class object
`...`	arguments for `print` function

Examples

```
data(applejuice)
print(applejuice)
```

`readEEM` *Read raw files and return a list*

Description

Read raw files from fluorescence spectrometer

Usage

```
readEEM(pathname = NULL)
```

Arguments

`pathname`	path to the files or folders which contains raw files (accept a vector).

Details

The supported format is outputs from FP-8500 (JASCO), F-7000 (Hitachi Hi-tech) and RF-6000 (Shimadzu) fluorescence spectrometer. It is likely that outputs from different machines of the same companies are supported by this function. Please send a word or pull request to add support for other formats.

Value

`readEEM` returns a list containing each raw files

`summary.EEM` *SummarizeEEM EEM list*

Description

Summarize by listing the sample number, names and their dimensions

Usage

```
## S3 method for class 'EEM'
summary(object, ...)
```

Arguments

`object`	a list containing EEM data as created by `readEEM` function.
`...`	arguments for `summary` function

Value

Text on console

Examples

```
data(applejuice)
summary(applejuice)
```

unfold *Unfold EEM list into a matrix*

Description

Unfold EEM list into a matrix with columns as variables (wavelength conditions) and rows as samples.

Usage

```
unfold(EEM)

## S3 method for class 'EEM'
unfold(EEM)
```

Arguments

EEM	a list containing EEM data as created by readEEM function.

Value

Unfolded EEM matrix where columns are wavelength condition and rows are samples

Examples

```
data(applejuice)
applejuice_uf <- unfold(applejuice) # unfold list into matrix
dim(applejuice_uf) # dimension of unfolded matrix
```

[.EEM *Subset EEM list*

Description

Subset EEM list

Usage

```
## S3 method for class 'EEM'
x[i, ...]
```

Arguments

x	EEM class object
i	indices specifying elements to extract
...	arguments for subset function

Examples

```
data(applejuice)
selected <- applejuice[1-5]
```

Index