

# Package ‘EEM’

January 20, 2025

**Type** Package

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

**Version** 1.1.1

**Date** 2016-04-21

**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, R.utils, sp, ggplot2

**Suggests** stats, pls, knitr, testthat

**VignetteBuilder** knitr

**License** GPL-3

**URL** <https://github.com/chengvt/EEM>

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-04-21 13:47:18

## Contents

applejuice . . . . .	2
commonizeEEM . . . . .	3
cutEEM . . . . .	3
delScattering . . . . .	4
delScattering2 . . . . .	5
drawEEM . . . . .	6
drawEEMgg . . . . .	7
EEM . . . . .	9

EEM-misc . . . . .	10
extract . . . . .	11
findLocalMax . . . . .	11
fold . . . . .	12
gluten . . . . .	13
normalize . . . . .	14
plotLoading . . . . .	15
plotReg . . . . .	16
plotScore . . . . .	16
plotScorem . . . . .	18
prcompname . . . . .	19
print.EEM . . . . .	20
readEEM . . . . .	20
summary.EEM . . . . .	21
unfold . . . . .	22
[.EEM . . . . .	22
<b>Index</b>	<b>24</b>

---

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)
```

---

commonizeEEM	<i>Smooth out the different dimensions of EEM data</i>
--------------	--

---

**Description**

Smooth out the difference dimensions of EEM data by finding the common variables of all data and subset those data.

**Usage**

```
commonizeEEM(EEM)
```

**Arguments**

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

**Value**

EEM class object with only common variables

**Examples**

```
data(gluten)
data(applejuice)
data <- c(gluten, applejuice)
summary(data) # different dimensions
data_combined <- commonizeEEM(data)
summary(data_combined) # same dimension, ready for unfold
```

---

cutEEM	<i>Cut portions of EEM</i>
--------	----------------------------

---

**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 <code>readEEM</code> function or EEMweight object generated by <code>extract</code> -related functions.
cutEX	Numeric or sequential data specifying regions to be cut for excitation wavelength. Examples, 200 or 200:500
cutEM	Numeric or sequential data specifying regions to be cut for emission wavelength. Examples, 200 or 200:500

**Value**

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

**Examples**

```
data(applejuice)
applejuice_cut <- cutEEM(applejuice, cutEX = 300:450)
drawEEM(applejuice_cut, 1)
```

---

delScattering	<i>Delete scattering rays</i>
---------------	-------------------------------

---

**Description**

This function deletes two regions that are not related to fluorescence emission: (1) regions where emission wavelength is shorter 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 = 40)
```

**Arguments**

EEM	A list containing EEM data as created by <code>readEEM</code> 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)
```

---

delScattering2	<i>Delete scattering rays</i>
----------------	-------------------------------

---

## Description

This function deletes three regions that are not related to fluorescence emission: (1) regions where emission wavelength is shorter than excitation light ( $E_m \leq E_x$ ), (2) scattering rays and their second order light, (3) regions above second-order scattering ( $EM \geq 2*EX$ )

## Usage

```
delScattering2(EEM, rep = 0, first = 30, second = 40)
```

## Arguments

EEM	A list containing EEM data as created by <a href="#">readEEM</a> 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]

## Value

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

## Examples

```
data(applejuice)
drawEEM(delScattering2(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, n, 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, n, 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

<code>x</code>	a list of EEM data generated by <code>readEEM</code> function or <code>EEMweight</code> object generated by <code>extract</code> -related functions.
<code>...</code>	(optional) further arguments passed to other methods of <code>filled.contour</code>
<code>n</code>	sample number. The number should not exceed <code>length(EEM)</code>
<code>exlab</code>	(optional) excitation-axis label
<code>emlab</code>	(optional) emission-axis label
<code>color.palette</code>	(optional) contour color palette. See <code>palette</code> 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 unfolded matrix which have column names in the format of EX...EM...
- data.frame: draw contour of unfolded data.frame which have column names in the format of EX...EM...
- numeric: draw contour of a vector of numeric values which have names in the format of EX...EM...

### See Also

[drawEEM](#)

### 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
```

---

drawEEMgg

*Draw contour for EEM data using ggplot2*

---

### Description

This function draw contour for EEM data using ggplot2. Use 'ggsave' to save the contours.

**Usage**

```
drawEEMgg(x, ...)

## S3 method for class 'EEM'
drawEEMgg(x, n, textsize = 20, color.palette = matlab.like,
  nlevels = 20, exlab = "Excitation wavelength [nm]",
  emlab = "Emission wavelength [nm]", main = NULL, has_legend = TRUE,
  zlim = NULL, breaks = waiver(), flipaxis = FALSE, ...)

## S3 method for class 'EEMweight'
drawEEMgg(x, ncomp, textsize = 25,
  color.palette = matlab.like, nlevels = 20,
  exlab = "Excitation wavelength [nm]", emlab = "Emission wavelength [nm]",
  main = NULL, has_legend = TRUE, zlim = NULL, breaks = waiver(),
  flipaxis = FALSE, ...)
```

**Arguments**

x	a list of EEM data generated by <a href="#">readEEM</a> function or EEMweight object generated by <a href="#">extract</a> -related functions.
...	arguments for other methods
n	sample number. The number should not exceed length(EEM)
textsize	(optional) text size
color.palette	(optional) contour color palette. See <a href="#">palette</a> for more details
nlevels	(optional) number of levels used to separate range of intensity value
exlab	(optional) excitation-axis label
emlab	(optional) emission-axis label
main	(optional) plot title
has_legend	logical value for legend
zlim	zlim = c(min, max)
breaks	breaks
flipaxis	(optional) flip axis
ncomp	number of components

**Details**

[drawEEM](#) is faster and should be used.

**Value**

A figure is returned on the graphic device

**Methods (by class)**

- EEM: draw EEM of EEM data created by [readEEM](#) function
- EEMweight: draw contours of the output from [getLoading](#) and [getReg](#).



**See Also**[drawEEM](#)**Examples**

```
## Not run:
require(EEM)
require(ggplot2)
data(applejuice)
drawEEMgg(applejuice, 1) # draw EEM of sample no.1
drawEEMgg(applejuice, 1, color.palette = cm.colors) # draw EEM of sample no.31 with different color
drawEEMgg(applejuice, 1, nlevels = 10) # change nlevels

# manually define legend values
drawEEMgg(applejuice, 1, breaks = seq(from = 1000, to = 6000, by = 1000))

# can be combined with other ggplot2 commands
# add point to the plot
drawEEMgg(applejuice, 1) + geom_point(aes(x = 350, y = 500), pch = 17, cex = 10)

# add grid line to the plot
drawEEMgg(applejuice, 1) + theme(panel.grid = element_line(color = "grey"),
panel.grid.major = element_line(colour = "grey"))

# add bg color
drawEEMgg(applejuice, 1, has_legend = FALSE) + geom_raster(aes(fill = value)) +
geom_contour(colour = "white")

## End(Not run)
```

---

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 multi-variate analysis

**Details**

The latest version and documentation can be found [here](#).

**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 <a href="#">palette</a> for more details
string	string or vector of strings
digits	integer indicating the number of decimal places ( <code>round</code> ) or significant digits ( <code>signif</code> ) 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: generate colors
- getEX: get EX value
- getEM: get EM value

---

extract	<i>Extract values from other models</i>
---------	---

---

**Description**

Extract values from other models

**Usage**

```
getLoading(x)
```

```
getReg(x)
```

**Arguments**

x                    output variable from `prcomp` or `pls` 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	<i>Find local maximum peaks</i>
--------------	---------------------------------

---

**Description**

Find local maximum peaks in EEM data

**Usage**

```
findLocalMax(data, ...)
```

```
## S3 method for class 'EEM'
```

```
findLocalMax(data, n, threshold = 0.7, showprint = TRUE, ...)
```

```
## S3 method for class 'matrix'
```

```
findLocalMax(data, n, threshold = 0.7, showprint = TRUE,
...)
```

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

### Arguments

data	EEM data generated by <a href="#">readEEM</a> function, unfolded EEM data generated by <a href="#">unfold</a> function or a vector of numeric values which have names in the format of EX...EM...
...	(optional) further arguments passed to other methods
n	sample number. The number should not exceed length(EEM).
threshold	threshold value in between 0 and 1. Lower the value to cover low peaks.
showprint	logical value whether to print out the results or not

### Value

return a character vector of peak names. If showprint = TRUE, it will also print a dataframe of indicating the value of local maximum peaks.

### Methods (by class)

- EEM: for EEM data created by [readEEM](#) function
- matrix: for unfolded EEM data created by [unfold](#) function
- numeric: for a vector of numeric values which have names in the format of EX...EM...

### Examples

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

applejuice_uf <- unfold(applejuice)
findLocalMax(applejuice_uf, 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 'data.frame'
fold(EEM_uf, name = NULL, ...)

## 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.
...	arguments for other methods
name	optional for data.frame input to specify the sample names

**Value**

EEM a list containing EEM/EEM data

**Methods (by class)**

- data.frame: fold unfolded data.frame

**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 <a href="#">prcomp</a> or <a href="#">pls</a> functions
ncomp	number of components
...	(optional) arguments for <a href="#">drawEEM</a> and <a href="#">filled.contour</a>

**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 <a href="#">plsr</a> function
ncomp	number of components
...	(optional) arguments for <a href="#">drawEEM</a> and <a href="#">filled.contour</a>

**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, group2 = NULL,
  cex = 1.5, cex.legend = 1, label = NULL, pos = 4, col = NULL,
  pch = NULL, legendlocation = "bottomright", legendoutside = FALSE,
  rightwhitespace = 0, ...)
```

**Arguments**

prcompResult	output object from <a href="#">prcomp</a> function
xPC	an integer indicating PC component on x-axis
yPC	an integer indicating PC component on y-axis
group	a vector of numeric, character or factor class separating the samples into groups. Correspond to point color.
group2	The second group, can be a vector of numeric, character or factor class separating the samples into groups. Correspond to point shape.
cex	(optional) size of points on graphs
cex.legend	(optional) size of fonts in legend
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 palette
pch	point type palette
legendlocation	(optional)location of legend on graph. Look up <a href="#">legend</a> 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
...	additional arguments for <a href="#">par</a>

**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) # plot PC1 vs PC2 score
plotScore(result, pch = 3, col = "blue") # change shape and color

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

# two groups
cultivar <- sapply(strsplit(names(applejuice), split = "-"), "[", 2)
plotScore(result, group = country, group2 = cultivar)

# make the points more transparent
## Not run:
require(scales)
plotScore(result, group = country, group2 = country, col = alpha(generateColor(2), 0.7))

## End(Not run)

```

---

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 <a href="#">prcomp</a> function
ncomp	maximum number of PC score to plot
group	a vector of numeric, character or factor class separating the samples into groups.

cex	(optional) size of points on graphs
col	point color palette
pch	point type palette
legendtitle	legend title
...	additional arguments to be passed on to <a href="#">pairs</a>

**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	<i>Create name for prcomp result</i>
------------	--------------------------------------

---

**Description**

Create name for [prcomp](#) result

**Usage**

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

**Arguments**

prcompResult	output value from <a href="#">prcomp</a> 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(path = NULL)
```

**Arguments**

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

**Details**

The supported format is \*.txt, \*.csv and \*.dat files from FP-8500 (JASCO), F-7000 (Hitachi Hi-tech), RF-6000 (Shimadzu) and Aqualog (Horiba) 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	<i>SummarizeEEM EEM list</i>
-------------	------------------------------

---

**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	<i>Unfold EEM list into a matrix</i>
--------	--------------------------------------

---

**Description**

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

**Usage**

```
unfold(EEM, replaceNA = TRUE)
```

**Arguments**

EEM	a list containing EEM data as created by readEEM function.
replaceNA	logical value whether to replace NA with 0

**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	<i>Subset EEM list</i>
-------	------------------------

---

**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

- \* **dataset**
  - applejuice, [2](#)
  - gluten, [13](#)
- \* **scattering**
  - delScattering, [4](#)
  - delScattering2, [5](#)
- [.EEM, [22](#)
- applejuice, [2](#)
- commonizeEEM, [3](#)
- cutEEM, [3](#)
- delScattering, [4](#)
- delScattering2, [5](#)
- drawEEM, [6](#), [7–9](#), [15](#), [16](#)
- drawEEMgg, [7](#)
- drawEEMgg\_internal (drawEEMgg), [7](#)
- EEM, [9](#)
- EEM-misc, [10](#)
- EEM-package (EEM), [9](#)
- extract, [4](#), [6](#), [8](#), [11](#)
- filled.contour, [6](#), [15](#), [16](#)
- findLocalMax, [11](#)
- fold, [12](#)
- generateColor (EEM-misc), [10](#)
- generatePoint (EEM-misc), [10](#)
- getEM (EEM-misc), [10](#)
- getEX (EEM-misc), [10](#)
- getLoading, [7](#), [8](#)
- getLoading (extract), [11](#)
- getReg, [7](#), [8](#)
- getReg (extract), [11](#)
- gluten, [13](#)
- legend, [17](#)
- normalize, [14](#)
- pairs, [19](#)
- palette, [6](#), [8](#), [10](#)
- par, [17](#)
- plotLoading, [15](#)
- plotReg, [16](#)
- plotScore, [16](#), [19](#)
- plotScorem, [17](#), [18](#)
- pls, [11](#), [15](#), [16](#)
- prcomp, [11](#), [15–19](#)
- prcompname, [19](#)
- print.EEM, [20](#)
- readEEM, [4–8](#), [12](#), [20](#)
- summary.EEM, [21](#)
- unfold, [12](#), [22](#)