Package 'staRdom'

August 30, 2019

Type Package

Title PARAFAC Analysis of EEMs from DOM

Version 1.0.18

Date 2019-08-30

Depends R (>= 3.5), ggplot2 (>= 2.2.1), eemR (>= 1.0.1), parallel (>= 3.5)

Description This is a user-friendly way to run a parallel factor (PARAFAC) analysis (Harshman, 1971) <doi:10.1121/1.1977523> on excitation emission matrix (EEM) data from dissolved organic matter (DOM) samples (Murphy et al., 2013) <doi:10.1039/c3ay41160e>. The analysis includes profound methods for model validation. Some additional functions allow the calculation of absorbance slope parameters and create beautiful plots.

License AGPL

Encoding UTF-8

LazyData true

Imports dplyr (>= 0.8.1), tidyr (>= 0.8.3), stringr (>= 1.4.0), pracma (>= 2.2.5), zoo (>= 1.8-6), tibble (>= 2.1.3), multiway (>= 1.0-6), GGally (>= 1.4), graphics (>= 3.5), doParallel (>= 1.0.14), drc (>= 3.0-1), foreach (>= 1.4.4), data.table (>= 1.12.2), matrixStats (>= 0.54.0), MBA(>= 0.0-9), cdom(>= 0.1.0), R.matlab(>= 3.6.2), readr(>= 1.3.1), gtools(>= 3.8)

Suggests plotly, xlsx, knitr, kableExtra, rmarkdown, knitcitations

RoxygenNote 6.1.1

VignetteBuilder knitr

URL https://cran.r-project.org/package=staRdom

BugReports https://github.com/MatthiasPucher/staRdom/issues

NeedsCompilation no

Author Matthias Pucher [aut, cre],
Daniel Graeber [aut, ctb],
Stefan Preiner [ctb],
Renata Pinto [ctb]

Maintainer Matthias Pucher <matthias.pucher@wcl.ac.at>

Repository CRAN

Date/Publication 2019-08-30 14:10:02 UTC

R topics documented:

`.eem_csv` *Import EEMs from generic csv files.*

Description

Import EEMs from generic csv files.

Usage

```
.eem_csv(file, col = "ex")
```

Arguments

`file`	path to file
`col`	either "ex" or "em", whatever wavelength is arranged in columns

Value

list with EEM data

`.trans_parafac` *Add data of a PARAFAC model derived from multiway from EEMs*

Description

Add data of a PARAFAC model derived from multiway from EEMs

Usage

```
.trans_parafac(parafac, em, ex, samples, comp, const, norm_factors)
```

Arguments

parafac	parafac model
em	emission wavelengths
ex	excitation wavelengths
samples	sample names
comp	number of components
const	constraints
norm_factors	factors to invert normalisation

Value

parafac model

---

absorbance_read — *Reading absorbance data from txt and csv files.*

Description

Reading absorbance data from txt and csv files.

Usage

```
absorbance_read(absorbance_path, order = TRUE, recursive = TRUE,
  dec = NULL, sep = NULL, verbose = FALSE, ...)
```

Arguments

absorbance_path	directory containing absorbance data files or path to single file. See details for format of absorbance data.
order	logical, data is ordered according to wavelength
recursive	read files recursive, include subfolders
dec	optional, either you set a decimal separator or the table is tested for . and ,
sep	optional, either you set a field separator or it is tried to be determined automatically
verbose	logical, provide more information
...	additional arguments that are passed on to fread.

Details

If absorbance_path is a directory, contained files that end on "csv" or "txt" are passed on to `read.table`. If the path is a file, this file is read. Tables can either contain data from one sample or from several samples in columns. The first column is considered the wavelength column. A multi-sample file must have sample names as column names. All tables are combined to one with one wavelength column and one column for each sample containing the absorbance data. Column and decimal separators are guessed from the supplied data. In some cases, this can lead to strange results. Plaese set ‘sep‘ and ‘dec‘ manually if you encounter any problems.

Value

A data frame containing absorbance data. An attribute "location" contains the filenames where each sample was taken from.

See Also

`fread`

Examples

```
absorbance_path <- system.file("extdata", "absorbance", package = "staRdom")
absorbance <- absorbance_read(absorbance_path, verbose = TRUE)
```

---

abs_blcor *Baseline correction for absorbance data*

Description

Baseline correction for absorbance data

Usage

```
abs_blcor(abs_data, wlrange = c(680, 700))
```

Arguments

`abs_data`	data.frame containing samples in columns, the column containing wavelengths must be named "wavelength"
`wlrange`	range of wavelengths that should be used for correction, absorbance mean in that range is subtracted from each value (sample-wise)

Value

data.frame

Examples

```
data(absorbance)
abs_data_cor <- abs_blcor(absorbance)
```

`abs_fit_slope` *Fit absorbance data to exponential curve. `drm` is used for the fitting process.*

Description

Fit absorbance data to exponential curve. `drm` is used for the fitting process.

Usage

```
abs_fit_slope(wl, abs, lim, l_ref = 350, control = drmc(errorm = FALSE,
  noMessage = TRUE), ...)
```

Arguments

`wl`	vector containing wavelengths
`abs`	vector containing absorption in m^-1
`lim`	vector containing lower and upper limits for wavelengths to use
`l_ref`	numerical. reference wavelength, default is 350, if set to NA l_ref is fitted
`control`	control parameters for drm, see `drmc`
`...`	parameters that are passed on to drm

Value

numeric exponential slope coefficient

See Also

`drm`

Examples

```
data(absorbance)
abs_fit_slope(absorbance$wavelength,absorbance$sample1,lim=c(350,400),l_ref=350)
```

abs_parms *Calculating slopes and slope ratios of a data frame of absorbance data.*

Description

Calculating slopes and slope ratios of a data frame of absorbance data.

Usage

```
abs_parms(abs_data, cuvle, unit = "absorbance", add_as = NULL,
  limits = list(c(275, 295), c(350, 400), c(300, 700)),
  l_ref = list(275, 350, 300), S = TRUE, lref = FALSE, p = FALSE,
  model = FALSE, Sint = FALSE, interval = 21, r2threshold = 0.8,
  cores = parallel::detectCores(logical = FALSE), verbose = FALSE)
```

Arguments

abs_data	data frame containing absorbance data.
cuvle	path length in cm
unit	unit of absorbance data: if "absorbance", absorbance data is multiplied by log(10) = 2.303 for slope calculations
add_as	additionally to a254 and a300, absorbance at certain wavelengths can be added to the table
limits	list with vectors containig upper and lower bounds of wavelengeth ranges to be fitted
l_ref	list with reference wavelengths, same length as limits
S	logical, include slope parameter in the table
lref	logical, include reference wavelength in the table
p	logical, include ps of the coefficients in the table
model	logical, include complete model in data frame
Sint	logical, wether the spectral curve is calculated interval-wise (cdom_spectral_curve)
interval	passed on to cdom_spectral_curve
r2threshold	passed on to cdom_spectral_curve
cores	number of cores to be used for parallel processing
verbose	logical, additional information is provided

Details

The absorbance data is a data frame with the first column called "wavelength" containg the wavelength. Each other column contains the data from one sample. You can use absorbance_read to read in appropriate data.

The following spectral parameters are calculated:

- $S_275-295$ slope between 275 and 295 nm calculated with nonlinear regression
- $S_350-400$ slope between 350 and 400 nm calculated with nonlinear regression
- $S_300-700$ slope between 275 and 295 nm calculated with nonlinear regression
- SR slope ratio, calculated by $S_275-295$/$S_350-400$
- E2:E3 ratio a_250/a_365
- E4:E6 ratio a_465/a_665
- a_254 absorbance at 254 nm
- a_300 absorbance at 300 nm

Depending on available wavelength range, values might be NA. Additionally other wavelength limits can be defined. The slope ratio might fail in this case. For further details please refer to Helm et al. (2008).

Value

A data frame containing the adsorption slopes and slope ratios in column, one line for each sample.

References

Helms, J., Kieber, D., Mopper, K. 2008. Absorption spectral slopes and slope ratios as indicators of molecular weight, source, and photobleaching of chromophoric dissolved organic matter. Limnol. Oceanogr., 53(3), 955–969 http://onlinelibrary.wiley.com/doi/10.4319/lo.2008.53.3.0955/pdf

Examples

```
data(absorbance)

a1 <- abs_parms(absorbance,cuvle=5, verbose = TRUE)
a2 <- abs_parms(absorbance,cuvle=5,l_ref=list(NA,NA,NA), lref=TRUE) # fit lref as well
```

`as.data.frame.eem` *Converting EEM data from class eem to data.frame.*

Description

Converting EEM data from class eem to data.frame.

Usage

```
## S3 method for class 'eem'
as.data.frame(x, row.names = NULL, optional = FALSE,
  gather = TRUE, ...)
```

Arguments

`x`	blabla
`row.names`	asfas
`optional`	ignored
`gather`	logical, says whether data.frame is returned with excitation wavelength as column names or as values of a column. If the data is gathered, the sample name is added as value in a calumn
`...`	ignored

Value

A data frame containing the EEM data.

Examples

```
data(eem_list)

as.data.frame(eem_list[[1]])
as.data.frame(eem_list[[1]],gather=FALSE)
```

A_missing   *Calculate the amount of each component for samples not involved in model building*

Description

Samples from an eemlist that were not used in the modelling process are added as entries in the A-modes. Values are calculated using fixed B and C modes in the PARAFAC algorithm. B and C modes can be provided via a previously calculated model or as matrices manually.

Usage

```
A_missing(eem_list, pfmodel = NULL,
  cores = parallel::detectCores(logical = FALSE), components = NULL,
  const = NULL, control = NULL, ...)
```

Arguments

`eem_list`	object of class eemlist with sample data
`pfmodel`	object of class parafac
`cores`	number of cores to use for parallel processing
`components`	optionally supply components to use manually, either as a variable of class parafac_components or as a list of variables of class parafac_components, if you do so,
`const`	optional constraints for model, just used, when components are supplied

`control`	optional constraint control parameters for model, just used, when components are supplied
`...`	additional arguments passed to eem_parafac

Details

This function can be used to calculate A modes (sample loadings) for samples that were previously excluded from the modelling process (e.g. outliers). Another way to use it would be a recombination of components from different models and calculating the according sample loadings. Expecially the later application is experimental and results have to be seen critically! Nevertheless, I decided to supply this function to stimulate some experiments on that and would be interested in your findings and feedback.

Value

object of class parafac

Examples

```
data(eem_list)
data(pf_models)

A_missing(eem_list,pf4[[1]])
```

`eem2array` *Data from an eemlist is transformed into an array*

Description

Data matrices from EEM are combined to an array that is needed for a PARAFAC analysis.

Usage

```
eem2array(eem_list)
```

Arguments

`eem_list`	object of class eemlist

Value

object of class array

Examples

```
data(eem_list)

eem2array(eem_list)
```

`eempf4analysis` *Create table of PARAFAC components and (optionally) EEM peaks and indices as well as absorbance slope parameters.*

Description

Please refer to eem_biological_index, eem_coble_peaks, eem_fluorescence_index, eem_biological_index and abs_parms for details on the certain values

Usage

```
eempf4analysis(pfmodel, eem_list = NULL, absorbance = NULL,
  cuvl = NULL, n = 4, export = NULL, ...)
```

Arguments

pfmodel	PARAFAC model where loadings of the components are extracted
eem_list	optional eemlist used for peak and indices calculation
absorbance	optional absorbance table used for absorbance slope parameter calculation
cuvl	optional cuvette length of absorbance data in cm
n	optional size of moving window in nm for data smoothing in advance of peak picking
export	optional file path of csv or txt table where data is exported
...	additional parameters passed to write.table

Value

data frame

Examples

```
data(eem_list)
data(pf_models)

results <- eempf4analysis(pfmodel = pf4[[1]],
                          eem_list = eem_list,
                          cuvl = 5, n = 4)
```

eempf_bindxc — *Combining extracted components of PARAFAC models*

Description

Combining extracted components of PARAFAC models

Usage

```
eempf_bindxc(components)
```

Arguments

components	list of parafac_components

Value

parafac_components

Examples

```
data(pf_models)
pfmodel <- pf4[[1]]
comps <- eempf_excomp(pfmodel,c(1,3))
comps2 <- eempf_excomp(pfmodel,c(4,6))
comps3 <- eempf_bindxc(list(comps, comps2))
```

eempf_compare — *Plot a set of PARAFAC models to compare the single components*

Description

Three plots are returned:

1. plot of numer of components vs. model fit
2. plot of different components as colour maps
3. plot of different components as peak lines

The plots are intended to help with a suitable number of components.

Usage

```
eempf_compare(pfres, ...)
```

Arguments

`pfres`	list of several objects of class parafac
`...`	arguments passe don to `eempf_fits` and `eempf_plot_comps`

Value

3 objects of class ggplot

See Also

`eempf_fits`, `eempf_plot_comps`

Examples

```
data(pf_models)

eempf_compare(pf4)
```

---

eempf_comps3D — *3D plots of PARAFAC components*

Description

Interactive 3D plots are created using plotly.

Usage

```
eempf_comps3D(pfmodel, which = NULL)
```

Arguments

`pfmodel`	object of class parafac
`which`	optional, if numeric selects certain component

Value

plotly plot

Examples

```
## Not run:
data(pf_models)

eempf_comps3D(pf4[[1]])

## End(Not run)
```

eempf_comp_load_plot *Plot components from a PARAFAC model*

Description

Additionally a bar plot with the amounts of each component in each sample is produced.

Usage

```
eempf_comp_load_plot(pfmodel, ...)
```

Arguments

pfmodel	object of class parafac
...	attributes passe don to ggeem

Value

ggplot

See Also

ggeem, eempf_load_plot

Examples

```
data(pf_models)

eempf_comp_load_plot(pf4[[1]])
```

eempf_comp_mat *Extract EEM matrix for single components determined in the PARAFAC analysis*

Description

The components of a PARAFAC analysis are extracted as a data frame

Usage

```
eempf_comp_mat(pfmodel, gather = TRUE)
```

Arguments

pfmodel	object of class parafac
gather	logical value whether excitation wavelengths are a column, otherwise excitation wavelengths are column names

Value

a list of class data frames

Examples

```
data(pf_models)

eempf_comp_mat(pf4[[1]])
```

---

eempf_comp_names *Extract names from PARAFAC model components*

Description

Extract names from PARAFAC model components

Usage

```
eempf_comp_names(pfmodel)
```

Arguments

pfmodel	parafac model

Value

vector of names or list of vecters of names

Examples

```
data(pf_models)
eempf_comp_names(pf4)

eempf_comp_names(pf4) <- c("A","B","C","D","E","F","G")

value <- list(c("A1","B1","C1","D","E","F","G"),
c("A2","B2","C","D","E","F","G"),
c("A3","B3","C","D","E","F","G"),
c("A4","B4","C","D","E","F","G"),
c("A5","B5","C","D","E","F","G5")
)

eempf_comp_names(pf4) <- value
eempf_comp_names(pf4)

ggeem(pf4[[1]])
```

`eempf_comp_names<-` *Set names of PARAFAC components*

Description

Set names of PARAFAC components

Usage

```
eempf_comp_names(pfmodel) <- value
```

Arguments

pfmodel	model of class parafac
value	character vector containing the new names for the components

Value

parafac model

Examples

```
data(pf_models)

eempf_comp_names(pf4) <- c("A","B","C","D","E","F","G")
```

`eempf_corcondia` *Calculate the core consistancy of an EEM PARAFAC model*

Description

This is basically a wrapper for `corcondia` that deals with the normalisation of the original data., Other than `corcondia`, the default dicisor = "core".

Usage

```
eempf_corcondia(pfmodel, eem_list, divisor = "core")
```

Arguments

pfmodel	PARAFAC model
eem_list	eemlist
divisor	divisor, please refer to `corcondia`

Value

numeric

Examples

```
## Not run:
# due to data limitation in package, example does not work with that data!

# eempf_corcondia(pfmodel,eem_list)


## End(Not run)
```

---

eempf_corplot — *Plot correlations of components in samples*

Description

A pair plot showing correlations between samples is created.

Usage

```
eempf_corplot(pfmodel, normalisation = FALSE, lower = list(continuous =
  "smooth"), mapping = aes(alpha = 0.2), ...)
```

Arguments

pfmodel	object of class parafac
normalisation	logical, whether normalisation is undone or not
lower	style of lower plots, see ggpairs
mapping	aesthetic mapping, see ggpairs
...	passed on to ggpairs

Value

object of class ggplot

See Also

ggpairs

Examples

```
data(pf_models)
eempf_corplot(pf4[[1]])
```

eempf_cortable — *Calculating correlations between the component loadings in all samples (C-Modes).*

Description

Calculating correlations between the component loadings in all samples (C-Modes).

Usage

```
eempf_cortable(pfmodel, normalisation = FALSE, method = "pearson", ...)
```

Arguments

`pfmodel`	results from a PARAFAC analysis, class parafac
`normalisation`	logical, whether normalisation is undone or not
`method`	method of correlation, passed to `cor`
`...`	passed on to `cor`

Value

matrix

Examples

```
data(pf_models)
eempf_cortable(pf4[[1]])
```

eempf_eemqual — *Calculating EEMqual which is an indicator of a PARAFAC model's quality*

Description

Calculating EEMqual which is an indicator of a PARAFAC model's quality

Usage

```
eempf_eemqual(pfmodel, eem_list, splithalf = NULL, ...)
```

Arguments

`pfmodel`	PARAFAC model
`eem_list`	EEM data as eemlist
`splithalf`	optionally, you can supplie available splithalf results from model to decrease computation time
`...`	additional arguments passed to splithalf

Value

data frame containing fit, corcondia, product of best TCCs from splithalf analysis, eemqual and splithalf models

References

Rasmus Bro, Maider Vidal, EEMizer: Automated modeling of fluorescence EEM data, Chemometrics and Intelligent Laboratory Systems, Volume 106, Issue 1, 2011, Pages 86-92, ISSN 0169-7439

Examples

```
data(eem_list)
data(pf_models)

pfmodel <- pf4[[1]]
eempf_eemqual(eem_list,pfmodel) # insuficient example data to run!
```

---

eempf_excomp *Extracting components of a PARAFAC model*

Description

Extracting components of a PARAFAC model

Usage

```
eempf_excomp(pfmodel, comps)
```

Arguments

pfmodel	parafac model
comps	vector with numbers of components to extract

Value

list

Examples

```
data(pf_models)
pfmodel <- pf4[[1]]
comps <- eempf_excomp(pfmodel,c(1,3))
```

eempf_export — *Create one table containing the PARAFAC models factors and optionally exporting it to csv or txt*

Description

Create one table containing the PARAFAC models factors and optionally exporting it to csv or txt

Usage

```
eempf_export(pfmodel, export = NULL, Fmax = TRUE, ...)
```

Arguments

pfmodel	PARAFAC model
export	file path to export table
Fmax	rescale modes so the A mode shows the maximum fluorescence
...	additional parameters passed to write.table

Value

data frame

Examples

```
data(pf_models)

factor_table <- eempf_export(pf4[[1]])
```

eempf_fits — *Fits vs. components of PARAFAC models are plotted*

Description

Fits vs. components of PARAFAC models are plotted

Usage

```
eempf_fits(pfres, ...)
```

Arguments

pfres	list of objects of class parafac
...	arguments passed on to ggplot

Value

object of class ggplot

Examples

```
data(pf_models)

eempf_fits(pf4)
```

---

eempf_leverage — *Calculate the leverage of each emission and excitation wavelength and each sample from a single PARAFAC model*

Description

Calculate the leverage of each emission and excitation wavelength and each sample from a single PARAFAC model

Usage

```
eempf_leverage(pfmodel)
```

Arguments

pfmodel	object of class parafac

Value

list of 3 named vectors (emission, excitation wavelengths and samples)

Examples

```
data(pf_models)

eempf_leverage(pf4[[1]])
```

eempf_leverage_data *Combine leverages into one data frame and add optional labels.*

Description

Combine leverages into one data frame and add optional labels.

Usage

```
eempf_leverage_data(cpl, qlabel = 0.1)
```

Arguments

cpl	leverage, outpout from eempf_leverage
qlabel	optional, quantile of which labels are shown (1 = all, 0 = no labels)

Value

data frame

Examples

```
data(pf_models)

leverage <- eempf_leverage(pf4[[1]])
lev_data <- eempf_leverage_data(leverage)
```

eempf_leverage_ident *Plot leverage of emission wavelengths, excitation wavelengths and samples.*

Description

Plot is interactive where you can select values with your mouse. A list of vectors is returned to remove this outliers in a further step from your samples. The labels to be shown can be selected by adding the quatile of samples with highest leverages to be labeled.

Usage

```
eempf_leverage_ident(cpl, qlabel = 0.1)
```

Arguments

cpl	leverage, outpout from eempf_leverage
qlabel	optional, quantile of which labels are shown (1 = all, 0 = no labels)

Value

list of three vectors containing the names of selected samples

See Also

eempf_leverage_plot

Examples

```
data(pf_models)

leverage <- eempf_leverage(pf4[[1]])
outliers <- eempf_leverage_ident(leverage)
```

---

eempf_leverage_plot *Plot leverage of emission wavelengths, excitation wavelengths and samples.*

Description

The labels to be shown can be selected by adding the quatile of samples with highest leverages to be labeled.

Usage

```
eempf_leverage_plot(cpl, qlabel = 0.1)
```

Arguments

cpl	leverage, outpout from eempf_leverage
qlabel	optional, quantile of which labels are shown (1 = all, 0 = no labels)

Value

ggplot

See Also

eempf_leverage_ident

Examples

```
data(pf_models)

leverage <- eempf_leverage(pf4[[1]])
eempf_leverage_plot(leverage)
```

eempf_load_plot — *Plot amount of each component in each sample as bar plot*

Description

Plot amount of each component in each sample as bar plot

Usage

```
eempf_load_plot(pfmodel)
```

Arguments

pfmodel	parafac model

Value

ggplot

Examples

```
data(pf_models)

eempf_load_plot(pf4[[1]])
```

eempf_mleverage — *Calculate the leverage of each emission and excitation wavelength and each sample from a list of PARAFAC models*

Description

Calculate the leverage of each emission and excitation wavelength and each sample from a list of PARAFAC models

Usage

```
eempf_mleverage(pfres_comps, ecdf = FALSE, stats = FALSE)
```

Arguments

pfres_comps	object of class parafac
ecdf	logical, transforme leverages to according empirical quantiles (ecdf)
stats	logical, whether means and standard deviations are calculated from leverages

Value

data frame containing leverages of wavelengths and samples for each model

Examples

```
data(pf_models)

eempf_mleverage(pf3)
```

`eempf_openfluor` *Write out PARAFAC components to submit to openfluor.org.*

Description

openfluor.org offers the possibility to compare your results to others, that were uploaded to the database. This functions writes out a txt containing the header lines and your components. Please open the file in an editor and fill in further information that cannot be covered by this function.

Usage

```
eempf_openfluor(pfmodel, file, Fmax = TRUE)
```

Arguments

`pfmodel`	PARAFAC model
`file`	string, path to outputfile. The directory must exist, the file will be created or overwritten if already present.
`Fmax`	rescale modes so the A mode shows the maximum fluorescence. As openfluor does not accept values above 1, this is a way of scaling the B and C modes to a range between 0 and 1.

Value

txt file

Examples

```
data(pf_models)
eempf_openfluor(pf4[[1]],file.path(tempdir(),"openfluor_example.txt"))
```

eempf_plot_comps *Plot all components of PARAFAC models*

Description

The components can be plottet in two ways: either as a colour map or as two lines (emission, excitation wavelengths) intersecting at the component maximum. If the list of provided models is named, these names are shown in the plot. Otherwise, the models are automatically named by "model#".

Usage

```
eempf_plot_comps(pfres, type = 1, names = TRUE, contour = FALSE, ...)
```

Arguments

pfres	list of PARAFAC models
type	1 for a colour map and 2 for em and ex wavelength loadings
names	logical, whether names of components should be written into the plot
contour	in case of 3 dimensional component plots, contours are added
...	arguments passed on to other functions

Value

object of class ggplot

Examples

```
data(pf_models)

eempf_plot_comps(pf4, type = 1)
eempf_plot_comps(pf4, type = 2)
eempf_plot_comps(list(pf4[[1]],pf4[[1]]), type=1)
```

eempf_plot_ssccheck *Plot results from an SSC check*

Description

Plot results from an SSC check

Usage

```
eempf_plot_ssccheck(ssccheck)
```

Arguments

`ssccheck`	outpout from `eempf_ssccheck`

Value

ggplot element

Examples

```
data(pf_models)

ssccheck <- eempf_ssccheck(pf3[1:3])
eempf_plot_ssccheck(ssccheck)
```

`eempf_reorder` *Reorder PARAFAC components*

Description

Reorder PARAFAC components

Usage

```
eempf_reorder(pfmodel, order, decreasing = FALSE)
```

Arguments

`pfmodel`	model of class parafac
`order`	vector containing desired new order or "em" or "ex" to reorder according to emission or excitation wavelengths of the peaks
`decreasing`	logical, whether components are reordered according to peak wvalengths in a decreasing direction

Value

parafac model

Examples

```
data(pf_models)
ggeem(pf4[[1]])

pf4r <- eempf_reorder(pf4[[1]],"ex")
ggeem(pf4r)
```

eempf_report *Create a html report of a PARAFAC analysis*

Description

Create a html report of a PARAFAC analysis

Usage

```
eempf_report(pfmodel, export, eem_list = NULL, absorbance = NULL,
  meta = NULL, metacolumns = NULL, splithalf = FALSE,
  shmodel = NULL, performance = FALSE, residuals = FALSE, spp = 5,
  ...)
```

Arguments

pfmodel	PARAFAC model
export	path to exported html file
eem_list	optional EEM data
absorbance	optional absorbance data
meta	optional meta data table
metacolumns	optional column names of metadata table
splithalf	optional logical, states whether split-half analysis should be included
shmodel	optional results from split-half analysis. If this data is not supplied but EEM data is available the split-half analysis is calculated on the creation of the report. Calculating the split-half analysis takes some time!
performance	calculating model performance: eempf_eemqual
residuals	logical, whether residuals are plotted in the report
spp	plots per page for loadgins and residuals plot
...	arguments to or from other functions

Value

TRUE if report was created

Examples

```
folder <- system.file("extdata/EEMs", package = "staRdom") # load example data
eem_list <- eem_read(folder, recursive = TRUE, import_function = eem_csv)

abs_folder <- system.file("extdata/absorbance", package = "staRdom") # load example data
absorbance <- absorbance_read(abs_folder)

metatable <- system.file("extdata/metatable_dreem.csv",package = "staRdom")
```

```
meta <- read.table(metatable, header = TRUE, sep = ",", dec = ".", row.names = 1)

checked <- eem_checkdata(eem_list, absorbance, metadata = meta,
metacolumns = "dilution", error = FALSE)

eem_names(eem_list)
pfm <- A_missing(eem_list,pf4[[1]])
eempf_report(pfm, export = "~/pf_report.html", eem_list = eem_list,
             absorbance = absorbance, meta = metatable, metacolumns = "dilution")
```

`eempf_rescaleBC` *Rescale B and C modes of PARAFAC model*

Description

B and C modes (emission and excitation wavelengths) are rescaled to RMS of value newscale. This is compensated in A mode (sample loadings).

Usage

```
eempf_rescaleBC(pfmodel, newscale = "Fmax")
```

Arguments

`pfmodel`	object of class parafac
`newscale`	If (default) newscale = "Fmax", each component will be scaled so the maximum of each component is 1. It is also possible to set a desired root mean-square for each column of the rescaled mode. Can input a scalar or a vector with length equal to the number of factors for the given mode.

Value

object of class parafac

See Also

`rescale`

Examples

```
data(pf_models)

new_pf <- eempf_rescaleBC(pf4[[1]])
```

eempf_residuals — *Calculate residuals of EEM data according to a certain model*

Description

Calculate residuals of EEM data according to a certain model

Usage

```
eempf_residuals(pfmodel, eem_list, select = NULL,
  cores = parallel::detectCores(logical = FALSE)/2)
```

Arguments

pfmodel	PARAFAC model of class parafac
eem_list	eemlist containing EEM data
select	character vector containing the names of the desired samples
cores	number of cores to use for parallel processing

Value

data frame with EEM residuals

Examples

```
data(eem_list)
data(pf_models)

eempf_residuals(pf4[[1]],eem_list)
```

eempf_residuals_plot — *Plot samples by means of whole sample, each single component and residuum*

Description

A raster of plots is created. Each column shows one sample. The top n rows show the n components from the model according their occurance in the certain samples. The second last row shows the residual, not covered by any component in the model and the last row shows the whole sample.

Usage

```
eempf_residuals_plot(pfmodel, eem_list, res_data = NULL, spp = 5,
  select = NULL, residuals_only = FALSE,
  cores = parallel::detectCores(logical = FALSE), contour = FALSE)
```

Arguments

`pfmodel`	object of class parafac containing the generated model
`eem_list`	object of class eemlist with all the samples that should be plotted
`res_data`	optional, data of sample residuals related to the model, output from `eempf_residuals`
`spp`	optional, samples per plot
`select`	optional, character vector of samples you want to plot
`residuals_only`	plot only residuals
`cores`	number of cores to use for parallel processing
`contour`	logical, states whether contours should be plotted

Details

eem_list may contain samples not used for modelling. Calculation is done by `A_missing`. This especially interesting if outliers are excluded prior modelling and should be evaluated again afterwards.

Value

several ggplot objects

Examples

```
data(eem_list)
data(pf_models)

eempf_residuals_plot(pf4[[1]],eem_list)
```

---

eempf_ssc — *Calculate the shift-and shape-sensitive congruence (SSC) between model components*

Description

The data variable pf_models can be supplied as list of PARAFAC models, output from a splithalf analysis or list of matrices Please see details of calculation in: U.J. Wünsch, R. Bro, C.A. Stedmon, P. Wenig, K.R. Murphy, Emerging patterns in the global distribution of dissolved matter fluorescence, Anal. Methods, 11 (2019), pp. 888-893

Usage

```
eempf_ssc(pf_models, tcc = FALSE, m = FALSE)
```

Arguments

pf_models	list of either PARAFAC models or component matrices
tcc	if set TRUE, TCC is returned instead
m	logical, if TRUE, emission and excitation SSCs or TCCs are combined by calculating the geometric mean

Value

(list of) tables containing SCCs between components

Examples

```
pf_models <- pf3[1:3]

sscs <- eempf_ssc(pf_models)
sscs

tcc <- eempf_ssc(pf_models, tcc = TRUE)
tcc
## mixed tcc (combine em and ex)
mtcc <- eempf_ssc(pf_models, tcc = TRUE, m = TRUE)
mtcc

## compare results from splithalf analysis
sh_sscs <- eempf_ssc(sh)

sh_sscs
## view diagonals only (components with similar numbers only)
lapply(sh_sscs, lapply, diag)
```

---

eempf_ssccheck — *Check SSCs between different models or initialisations of one model*

Description

Check SSCs between different models or initialisations of one model

Usage

```
eempf_ssccheck(pfmodels, best = length(pfmodels), tcc = FALSE)
```

Arguments

pfmodels	list of parafac models
best	number of models with the highest R^2 to be used, default is all models
tcc	logical, if TRUE, TCC instead of SSC is calculated

Value

data.frame containing SSCs

Examples

```
data(pf_models)

eempf_ssccheck(pf3[1:2])


# SSCs of split-half models, models need to be unlisted
data(sh)
eempf_ssccheck(unlist(sh, recursive = FALSE))
```

---

eempf_varimp — *Calculate the importance of each component.*

Description

Calculate the importance of each component.

Usage

```
eempf_varimp(pfmodel, eem_list, cores = parallel::detectCores(logical =
  FALSE), ...)
```

Arguments

`pfmodel`	model of class parafac
`eem_list`	eemlist used to calculate that model
`cores`	cores to be used for the calculation
`...`	other aruments passed to eem_parafac

Details

The importance of each variable is calculated by means of creating a model without a specific component and calculating the difference between the original R-squared and the one with the left out component. The derived values state the loss in model fit if one component is not used in the modeling process. For the creation of the new models, the exact components of the original model are used.

Value

numeric vector, values are in the same order of the components in the supplied model.

Examples

```
data(pfmodel)
data(eem_list)

eempf_varimp(pf4[[1]],eem_list)
```

---

eem_absdil *Multiply absorbance data according to the dilution and remove absorbance from samples where undiluted data is used.*

Description

According to dilution data absorbance is either multiplied by the according factor or the undiluted absorbance data is deleted. You can either specify the cor_data data table coming from `eem_dilcorr` or supply an eemlist, and the dilution data to created on the fly.

Usage

```
eem_absdil(abs_data, eem_list = NULL, dilution = NULL,
  cor_data = NULL, auto = TRUE, verbose = FALSE)
```

Arguments

`abs_data`	absorbance data
`eem_list`	optional eemlist
`dilution`	optional dilution data as data frame
`cor_data`	optional output from `eem_dilcorr` as data frame
`auto`	optional, see `eem_dilcorr`
`verbose`	optional, see `eem_dilcorr`

Value

data frame

Examples

```
# no appropriate exmaple data available yet
```

eem_checkdata *Check your EEM, absorption and metadata before processing*

Description

The function tries to lead you to possible problems in your data.

Usage

```
eem_checkdata(eem_list, absorbance, metadata = NULL,
  metacolumns = NULL, correction = FALSE, error = TRUE)
```

Arguments

eem_list	eemlist continaing EEM data.
absorbance	data.frame containing absorbance data.
metadata	optional data.frame containing metadata.
metacolumns	character vector of columns that are checkt for complete data sets
correction	logical, whether EEMs should be checked for applied corrections
error	logical, whether a problem should cause an error or not.

Details

The returned list contains character vectors with sample names where possible problems were found: problem (logical, whether a severe problem was found), nas (sample names with NAs in EEM data), missing_correction (correction of EEM samples was not done or not done successfully),eem_no_abs (EEM samples with no absorbance data), abs_no_eem (samples with present absorbance but no EEM data), duplse (duplicate sample names in EEM data), duplsa (duplicate sample names in absorbance data), invalid_eem (invalid EEM sample name), invalid_abs (invalid absorbance sample name), range_mismatch (wavelength ranges of EEM and absorbance data are mismatching), metadupls (duplicate sample names in metadata), metamissing (EEM samples where metadata is missing), metaadd (samples in metadata without EEM data)

Value

writes out possible porblems to command line, additionally list with sample names where possible problems were found, see details.

Examples

```
folder <- system.file("extdata/EEMs", package = "staRdom") # load example data
eem_list <- eem_read(folder, recursive = TRUE, import_function = eem_csv)

abs_folder <- system.file("extdata/absorbance", package = "staRdom") # load example data
absorbance <- absorbance_read(abs_folder)

metatable <- system.file("extdata/metatable_dreem.csv",package = "staRdom")
```

```
meta <- read.table(metatable, header = TRUE, sep = ",", dec = ".", row.names = 1)

checked <- eem_checkdata(eem_list, absorbance, metadata = meta,
metacolumns = "dilution", error = FALSE)
# This example returns a message, that absorbance data for the
# blank samples are missing. As absorbance is supposed to be 0 over
# the whole spectrum when you measure blanks, there is no need
# to supply the data and do an inner-filter effect correction.
```

---

eem_checksize *Check size of EEMs*

Description

The size of EEMs in an eemlist is checked and the sample names of samples with more data than the sample with the smallest range are returned.

Usage

```
eem_checksize(eem_list)
```

Arguments

eem_list	eemlist

Value

character vector

Examples

```
data(eem_list)
eem_checksize(eem_list)
```

---

eem_corrections *Return names of samples where certain corrections are missing.*

Description

Return names of samples where certain corrections are missing.

Usage

```
eem_corrections(eem_list)
```

Arguments

eem_list	eemlist to be checked

Value

prints out sample names

Examples

```
data(eem_list)

eem_corrections(eem_list)
```

---

eem_csv *Importer function for generic csv files to be used with eem_read().*

Description

This function can be used to import generic csv files containing EEM data using eem_read. Excitation wavelengths are assumed column-wise and emission wavelengths row-wise. If your data is arranged the other way round, please use eem_csv2

Usage

```
eem_csv(file)
```

Arguments

file	path to file passed from eem_read

Value

list with EEM data

Examples

```
eems <- system.file("extdata/EEMs",package="staRdom")
eem_list <- eem_read(eems, recursive = TRUE, import_function = eem_csv)

eem_list
```

eem_csv2 — *Importer function for generic csv files to be used with eem_read().*

Description

This function can be used to import generic csv files containing EEM data using eem_read. Excitation wavelengths are assumed row-wise and emission wavelengths column-wise If your data is arranged the other way round, please use eem_csv

Usage

```
eem_csv2(file)
```

Arguments

file	path to file passed from eem_read

Value

list with EEM data

Examples

```
## no example data provided with the package
## below is an example how this could like like
eems <- "C:/some/path/to/eem.csv"
eem_list <- eem_read(eems, recursive = TRUE, import_function = eem_csv2)

eem_list
```

eem_dilcorr — *Create table how samples should be corrected because of dilution*

Description

Due to dilution absorbance spectra need to be multiplied by the dilution factor and names of EEM samples can be adjusted to be similar to their undiluted absorbance sample. The table contains information about these two steps. Undiluted samples are suggested by finding absorbance samples match the beginning of EEM sample name (see details).

Usage

```
eem_dilcorr(eem_list, abs_data, dilution, auto = FALSE, verbose = TRUE)
```

Arguments

eem_list	eemlist
abs_data	absorbance data as data frame
dilution	dilution data as data frame with rownames
auto	way how to deal with dilution is chosen automatically. See details.
verbose	print out more information

Details

If you choose an automatic analysis EEMs are renamed if there is only one matching undiluted absorbance sample. Matching samples is done by comparing the beginning of the sample name (e.g. "sample3_1to10" fits "sample3").

Value

data frame

Examples

```
# no appropriate exmaple data available yet
```

---

eem_dilution *Modifying fluorescence data according to dilution.*

Description

If samples were diluted before measuring, a dilution factor has to be added to the measured data. This function can do that by either multilpying each sample with the same value or using a data frame with different values for each sample.

Usage

```
eem_dilution(data, dilution = 1)
```

Arguments

data	fluorescence data with class eemlist
dilution	dilution factor(s), either numeric value or data frame. Row names of data frame have to be similar to sample names in eemlist.

Value

fluorescence data with class eemlist

Examples

```
data(eem_list)

eem_list <- eem_dilution(eem_list,dilution=5)
```

`eem_duplicates` *Check for duplicate sample names*

Description

Check for duplicate sample names

Usage

```
eem_duplicates(data)

## Default S3 method:
eem_duplicates(data)

## S3 method for class 'eemlist'
eem_duplicates(data)

## S3 method for class 'data.frame'
eem_duplicates(data)
```

Arguments

`data`	eemlist or data.frame containing absorbance data

Value

named character vector with duplicate sample names

Examples

```
### check
```

eem_easy *Opens an R markdown template for an easy and userfriendly analysis of EEM data.*

Description

In your default editor (e.g. RStudio), a Rmd file is opened. It consists of bloacks gathering the parameters and information needed and continues with a series of data corrections, peak picking and plots. Finally you get a report of your analysis, a table with the peaks and optional pngs of your fluorescence data. To continue working and keeping your settings, the file can be sa ved anywhere and reused anytime.

Usage

```
eem_easy()
```

Details

Function does not work well in Windows. You might try file.edit(system.file("EEM_simple_analysis.Rmd", package = "staRdom"))

Value

A pdf report, a peak picking table and optional plots.

Examples

```
## Not run:
#
eem_easy()

# this function fails very often, so you might use that:
file.edit(system.file("EEM_simple_analysis.Rmd", package = "staRdom"))

## End(Not run)
```

eem_eemdil *Correct names of EEM samples to match undiluted absorbance data.*

Description

Correct names of EEM samples to match undiluted absorbance data.

Usage

```
eem_eemdil(eem_list, abs_data = NULL, dilution = NULL,
  cor_data = NULL, auto = TRUE, verbose = FALSE)
```

Arguments

`eem_list`	eemlist
`abs_data`	optinal absorbance data as data frame
`dilution`	optinal dilution data as data frame
`cor_data`	optional output from `eem_dilcorr` as data frame
`auto`	optional, see `eem_dilcorr`
`verbose`	optional, see `eem_dilcorr`

Value

eemlist

Examples

```
# no appropriate exmaple data available yet
```

---

eem_exclude — *Exclude complete wavelengths or samples form data set*

Description

Outliers in all modes should be avoided. With this functions excitation or emission wavelengths as well as samples can be removed completely from your sample set.

Usage

```
eem_exclude(eem_list, exclude = list, verbose = FALSE)
```

Arguments

`eem_list`	object of class eemlist
`exclude`	list of three vectors, see details
`verbose`	states whether additional information is given in the command line

Details

The argument exclude is a named list of three vectors. The names must be "ex", "em" and "sample". Each element contains a vector of wavelengths or sample names that are to be excluded from the data set.

Value

object of class eemlist

Examples

```
data(eem_list)

exclude <- list("ex" = c(280,285,290,295),
"em" = c(),
"sample" = c("667sf", "494sf")
)

eem_list_ex <- eem_exclude(eem_list, exclude)
```

---

eem_extend2largest *EEM sample data is extended to include all wavelengths in all samples*

Description

Compared to the whole sample set, wavelengths missing in some samples are added and set NA or interpolated. This can be especially helpful, if you want to combine data measured with different wavelength intervals in a given range.

Usage

```
eem_extend2largest(eem_list, interpolation = FALSE, ...)
```

Arguments

eem_list	eemlist
interpolation	logical, whether added NAs should be interpolated
...	arguments passed to eem_interp

Value

eemlist

Examples

```
library(dplyr)
data(eem_list)
eem_list <- eem_exclude(eem_list[1:5] %>%
`class<-`("eemlist"), exclude = list(em = c(318,322,326,550,438), ex = c(270,275))) %>%
eem_bind(eem_list[6:15] %>% `class<-`("eemlist"))
ggeem(eem_list)

eem_extend2largest(eem_list) %>%
  ggeem()
```

eem_getextreme — *Determines the the biggest range of EEM spectrum where data is available from each sample.*

Description

Determines the the biggest range of EEM spectrum where data is available from each sample.

Usage

```
eem_getextreme(data)
```

Arguments

data	eemlist

Value

list of numeric vector containing the biggest available range

Examples

```
data(eem_list)
eem_getextreme(eem_list)

eem_list <- eem_range(eem_list,ex = c(250,Inf),em = c(280,500))
eem_getextreme(eem_list)
```

eem_hitachi — *Importer function for Hitachi F-7000 txt files to be used with eem_read().*

Description

This function can be used to import txt files from Hitachi F-7000 containing EEM data using eem_read.

Usage

```
eem_hitachi(file)
```

Arguments

file	path to file passed from eem_read

Value

list with EEM data

Examples

```
## no example data provided with the package
## below is an example how this could like like
eems <- "C:/some/path/to/hitachi.TXT"
eem_list <- eem_read(eems, recursive = TRUE, import_function = eem_hitachi)

eem_list
```

---

eem_ife_correction *Wrapper function to allow eem_inner_filter_effect (eemR) handling different cuvette lengths.*

Description

Calls eem_inner_filter_effect for each sample to use different cuvette lengths.

Usage

```
eem_ife_correction(data, abs_data, cuvl, unit = "absorbance")
```

Arguments

data	fluorescence data of class eemlist
abs_data	absorbance data
cuvl	length of cuvette of absorption measurment in cm. Either a number or a data frame. Row names of data frame have to be similar to sample names in data
unit	unit of absorbance data. Either "absorbance" or "absorption".

Value

fluorescence data of class eemlist

Examples

```
folder <- system.file("extdata/cary/scans_day_1", package = "eemR") # load example data
eem_list <- eem_read(folder, import_function = "cary")
data(absorbance)

eem_ife_correction(eem_list, absorbance, 5, unit = "absorbance")
```

eem_import_dir — *Load all eemlist obects saved in different Rdata or RDa files in a folder.*

Description

Reads Rdata and RDa files with one eemlist each. The eemlists are combined into one and returned.

Usage

```
eem_import_dir(dir)
```

Arguments

dir	folder where RData files are saved

Value

eemlist

Examples

```
## Not run:
# due to package size issues no example data is provided for this function
# eem_import_dir("C:/some_folder/with_EEMS/only_Rdata_files")

## End(Not run)
```

eem_interp — *Missing values are interpolated within EEM data*

Description

Missing EEM data can be interpolated. Usually it is the result of removing scatter or other parts where noise is presumed. Different interpolation algorithms can be used (see details).

Usage

```
eem_interp(data, cores = parallel::detectCores(logical = FALSE),
  type = TRUE, verbose = FALSE, nonneg = TRUE, extend = FALSE, ...)
```

Arguments

data	object of class eemlist with spectra containing missing values
cores	specify number of cores for parallel computation
type	numeric 0 to 4 or TRUE which resembles type 1
verbose	logical, whether more information on calculation should be provided
nonneg	logical, whether negative values should be replaced by 0
extend	logical, whether data is extrapolated using type 1
...	arguments passed on to other functions (pchip, na.approx, mba.points)

Details

The types of interpolation are (0) setting all NAs to 0, (1) spline interpolation with mba.points, (2) excitation and emission wavelength-wise interpolation with pchip and subsequent mean, (3) excitation wavelength-wise interpolation with pchip and (4) linear interpolation in 2 dimensions with na.approx and again subsequent mean calculation. Calculating the mean is a way of ensuring NAs are also interpolated where missing boundary values would make that impossible. Using type = 1, extrapolation can be suppressed by adding the argument extend = FALSE.

Value

object of class eemlist with interpoleted spectra.

References

Elcoroaristizabal, S., Bro, R., García, J., Alonso, L. 2015. PARAFAC models of fluorescence data with scattering: A comparative study. Chemometrics and Intelligent Laboratory Systems, 142, 124-130 https://doi.org/10.1016/j.chemolab.2015.01.017

See Also

pchip, mba.points, na.approx

Examples

```
data(eem_list)
eem_list <- eem_list[1:6]
class(eem_list) <- "eemlist"

remove_scatter <- c(TRUE, TRUE, TRUE, TRUE)

remove_scatter_width = c(15,10,16,12)

eem_list <- eem_rem_scat(eem_list,remove_scatter,remove_scatter_width)

eem_list <- eem_interp(eem_list)

ggeem(eem_list)
```

```
eem_list2 <- eem_setNA(eem_list,ex=200:280,interpolate=FALSE)

ggeem(eem_list2)

eem_list3 <- eem_interp(eem_list2,type=1,extend = TRUE)

ggeem(eem_list3)

eem_list3 <- eem_interp(eem_list2,type=1,extend = FALSE)

ggeem(eem_list3)
```

---

eem_is.na *Check for NAs in EEM data*

Description

Check for NAs in EEM data

Usage

```
eem_is.na(eem_list)
```

Arguments

eem_list	eemlist to check

Value

named character vector with sample names where EEM data contains NAs

Examples

```
### check
```

`eem_list` *15 fluorescence samples from drEEM used for examples.*

Description

15 fluorescence samples from drEEM used for examples.

Usage

```
eem_list
```

Format

eemlist

`eem_list_outliers` *2 fluorescence samples from drEEM that were excluded as outliers from the PARAFAC model.*

Description

2 fluorescence samples from drEEM that were excluded as outliers from the PARAFAC model.

Usage

```
eem_list_outliers
```

Format

eemlist

`eem_load_dreem` *Load original data from the drEEM tutorial and return it as eemlist*

Description

Load original data from the drEEM tutorial and return it as eemlist

Usage

```
eem_load_dreem()
```

Value

eemlist

Examples

```
eem_list <- eem_load_dreem()
```

---

eem_matmult *Multiply all EEMs with a matrix*

Description

Multiply all EEMs with a matrix

Usage

```
eem_matmult(eem_list, matrix = NULL, value = 0)
```

Arguments

`eem_list`	EEM data as eemlist
`matrix`	either a vactor containing "l" and/or "u" or a matrix, see details.
`value`	in case matrices "l" or "u" are used, this specifies the value to use in this areas. Usually this is 0 (default) or NA but any numeric value can be used.

Details

All EEMs must be of the same size. If matrix is of type matrix, it is used right away to multiply the EEMs. It has to be of the same size as the EEMs. If matrix is a vector containing "l", values below 1st order Rayleigh scattering are set to 0. If matrix contains "u", values above 2nd order Raman scattering are set to 0. If you want to remove wavelength ranges, take into consideration to use `eem_cut` or `eem_range`.

Value

eemlist

Examples

```
data(eem_list)
eem <- eem_list[1:9]
class(eem) <- "eemlist"

ggeem(eem)

eem_list_cut <- eem_matmult(eem,matrix=c("l"), value= NA)
ggeem(eem_list_cut)
```

eem_metatemplate *Create table that contains sample names and locations of files.*

Description

You can use this table as an overview of your files and/or as a template for creating a metadata table.

Usage

```
eem_metatemplate(eem_list = NULL, absorbance = NULL)
```

Arguments

eem_list	eemlist
absorbance	data frame with absorbance data

Value

data frame

Examples

```
folder <- system.file("extdata/EEMs", package = "staRdom") # load example data
eem_list <- eem_read(folder, recursive = TRUE, import_function = eem_csv)
data(absorbance)

eem_metatemplate(eem_list,absorbance)
```

eem_name_replace *Replace matched patterns in sample names*

Description

Sample names in eemlist can be altered.

Usage

```
eem_name_replace(eem_list, pattern, replacement)
```

Arguments

eem_list	data of class eemlist
pattern	character vector containing pattern to look for.
replacement	character vector of replacements. Has to have the same length as pattern

Details

str_replace_all from package stringr is used for the replacement. Please read the corresponding help for further options.

Value

An eemlist.

See Also

str_replace_all

Examples

```
eem_names(eem_list)

eem_list <- eem_name_replace(eem_list,"sample","Sample")
eem_names(eem_list)
```

---

eem_overview_plot — *Plot fluorescence data from several samples split into several plots.*

Description

Plot fluorescence data from several samples split into several plots.

Usage

```
eem_overview_plot(data, spp = 8, ...)
```

Arguments

data	fluorescence data of class eemlist
spp	number of samples per plot
...	arguments passed on to ggeem

Value

list of ggplots

Examples

```
data(eem_list)
eem_overview_plot(eem_list,spp=9)
```

eem_parafac *Runs a PARAFAC analysis on EEM data*

Description

One or more PARAFAC models can be calculated depending on the number of components. The idea is to compare the different models to get the most suitable. B-mode is emmission wavelengths, C-mode is excitation wavelengths and, A-mode is the loadings of the samples. The calculation is done with parafac, please see details there.

Usage

```
eem_parafac(eem_list, comps, maxit = 2500, normalise = TRUE,
  const = c("nonneg", "nonneg", "nonneg"), nstart = 20, ctol = 10^-8,
  strictly_converging = FALSE, cores = parallel::detectCores(logical =
  FALSE), verbose = FALSE, output = "best", ...)
```

Arguments

eem_list	object of class eem
comps	vector containing the desired numbers of components. For each of these numbers one model is calculated
maxit	maximum iterations for PARAFAC algorithm
normalise	state whether EEM data should be normalised in advance
const	constraints of PARAFAC analysis. Default is non-negative ("nonneg"), alternatively smooth and non-negative ("smonon") might be interesting for an EEM analysis.
nstart	number of random starts
ctol	Convergence tolerance (R^2 change)
strictly_converging	calculate nstart converging models and take the best. Please see details!
cores	number of parallel calculations (e.g. number of physical cores in CPU)
verbose	print infos
output	Output the "best" solution (default) only or additionally add "all" nstart solutions to the model as an element named "models".
...	additional parameters that are passed on to parafac

Details

PARAFAC models are created based on multiple random starts. In some cases, a model does not converge and the resulting model is then based on less than nstart converging models. In case you want to have nstart converging models, set strictly_converging TRUE. This calculates models stepwise until the desired number is reached but it takes more calculation time. Increasing the number of models from the beginning is much more time efficient.

Value

object of class parafac

See Also

parafac

Examples

```
data(eem_list)

dim_min <- 3 # minimum number of components
dim_max <- 7 # maximum number of components
nstart <- 25 # random starts for PARAFAC analysis, models built simulanuously, best selected
cores <- parallel::detectCores(logical=FALSE) # use all cores but do not use all threads
maxit = 2500
ctol <- 10^-7 # tolerance for parafac

pfres_comps <- eem_parafac(eem_list, comps = seq(dim_min, dim_max),
    normalise = TRUE, maxit = maxit, nstart = nstart, ctol = ctol, cores = cores)

pfres_comps2 <- eem_parafac(eem_list, comps = seq(dim_min, dim_max),
   normalise = TRUE, maxit = maxit, nstart = nstart, ctol = ctol, cores = cores, output = "all")
```

---

eem_raman_area *Calculate raman area of EEM samples*

Description

Calculate raman area of EEM samples

Usage

```
eem_raman_area(eem_list, blanks_only = TRUE, average = FALSE)
```

Arguments

eem_list	An object of class eemlist.
blanks_only	logical. States whether all samples or just blanks will be used.
average	logical. States whether samples will be averaged before calculating the raman area.

Details

Code based on eem_raman_normalisation.

Value

data frame containing sample names, locations and raman areas

Examples

```
folder <- system.file("extdata/EEMs",package="staRdom")
eem_list <- eem_read(folder, recursive = TRUE, import_function = eem_csv)
blank <- eem_extract(eem_list,sample ="blank", keep = TRUE)

eem_raman_area(blank)
```

`eem_raman_normalisation2`

Wrapper function to eem_raman_normalisation (eemR).

Description

Usually Raman normalisation is done with fluorescence data from a blank sample. Sometimes you already know a value for the Raman area. This function can do both.

Usage

```
eem_raman_normalisation2(data, blank = "blank")
```

Arguments

`data`	fluorescence data of class eemlist
`blank`	defines how Raman normalisation is done (see 'Details')

Details

Possible values for blank:

"blank": normalisation is done with a blank sample. Please refer to `eem_raman_normalisation`.

numeric: normalisation is done with one value for all samples.

data frame: normalisation is done with different values for different samples. Values are taken from a data.frame with sample names as rownames and one column containing the raman area values.

Value

fluorescence data of class eemlist

Examples

```
data(eem_list)
# correction by blank
eems_bl <- eem_raman_normalisation2(eem_list,blank="blank")

# correction by value
eems_num <- eem_raman_normalisation2(eem_list,blank=168)
```

`eem_range` *Cut EEM data matching a given wavelength range*

Description

Cut EEM data matching a given wavelength range

Usage

```
eem_range(data, ex = c(0, Inf), em = c(0, Inf))
```

Arguments

data	EEM data as eemlist
ex	optional desired range of excitation wavelength
em	optional desired range of emission wavelength

Value

An eemlist of reduced spectra size.

Examples

```
data(eem_list)
eem_range(eem_list,ex = c(250,Inf),em = c(280,500))
```

`eem_read_csv` *Import EEMs from generic csv tables (deprecated)*

Description

This function is deprecate, please use `eem_read`(..., import_function = `eem_csv`) or eem_read(..., import_function = `eem_csv2`) instead. EEM data is loaded from generic files. First column and first row contains wavelength values. The other values are to be plain numbers. `fread` is used to read the table. It offers a lot of helpful functions (e.g. skipping any number n of header lines by adding ‘skip = n‘)

Usage

```
eem_read_csv(path, col = "ex", recursive = TRUE,
  is_blank_corrected = FALSE, is_scatter_corrected = FALSE,
  is_ife_corrected = FALSE, is_raman_normalized = FALSE,
  manufacturer = "unknown", ...)
```

Arguments

`path`	path to file(s), either a filename or a folder
`col`	either "ex" or "em", what wavelengths are in the columns
`recursive`	logical, whether directories are loaded recursively
`is_blank_corrected`	logical, whether blank correction was done
`is_scatter_corrected`	logical, wether scatters were corrected
`is_ife_corrected`	logical, wether inner-filter effect correction was done
`is_raman_normalized`	logical, wether raman normalisation applied
`manufacturer`	string specifying manufacturer of instrument
`...`	parameters from other functions, currently not used

Examples

```
eems <- system.file("extdata/EEMs",package="staRdom")
eem_list <- eem_read_csv(eems)

eem_list
```

---

eem_red2smallest — *Remove wavelengths, that are missing in at least one sample form the whole set.*

Description

Remove wavelengths, that are missing in at least one sample form the whole set.

Usage

```
eem_red2smallest(data, verbose = FALSE)
```

Arguments

`data`	data of EEM samples as eemlist
`verbose`	states whether additional information is given in the command line

Details

This step is neccessary to perform a PARAFAC analysis which can only be calculated with spectra of similar range.

Value

eemlist with reduced spectral width

Examples

```
require(dplyr)

data(eem_list)

eem_list_red <- eem_red2smallest(eem_list)

# create an eemlist where data is missing
eem_list2 <- eem_exclude(eem_list,
    list("ex" = c(280,290,350),
         "em" = c(402,510),
         "sample" = c()))

# modify names of samples with missing data
eem_names(eem_list2) <- paste0("x",eem_names(eem_list2))

# combined the lists with and without missing data
eem_list3 <- eem_bind(eem_list,eem_list2)
#ggeem(eem_list3)

# reduce the data in the whole sampleset to the smallest wavelengths that are present in all samples
eem_list4 <- eem_red2smallest(eem_list3)
#ggeem(eem_list4)
```

`eem_rem_scat` *Remove Raman and Rayleigh scattering in fluorescence data*

Description

Wrapper function to remove several scatterings in one step using `eem_remove_scattering`.

Usage

```
eem_rem_scat(data, remove_scatter, remove_scatter_width = 10,
  interpolation = FALSE, cores = parallel::detectCores(logical =
  FALSE), verbose = FALSE)
```

Arguments

data	object of class eemlist
remove_scatter	logical vector. The meanings of the vector are "raman1", "raman2", "rayleigh1" and "rayleigh2" scattering. Set TRUE if certain scattering should be removed.
remove_scatter_width	numeric vector containing width of scattering to remove. If there is only one element in this vector, each this is the width of each removed scattering. If there are 4 values, differnt widths are used ordered by "raman1", "raman2", "rayleigh1" and "rayleigh2".
interpolation	logical, optionally states whether interpolation is done right away
cores	optional, CPU cores to use for interpolation
verbose	logical, provide additional information

Value

eemlist

Examples

```
data(eem_list)

remove_scatter <- c(TRUE, TRUE, TRUE, TRUE)

remove_scatter_width = c(15,10,16,12)

eem_rem_scat(eem_list,remove_scatter,remove_scatter_width)
```

---

eem_scale_ext *Determine the range of fluorescence values in a set of samples*

Description

Determine the range of fluorescence values in a set of samples

Usage

```
eem_scale_ext(data)
```

Arguments

data	eemlist containing the EEM data

Value

numeric vector

Examples

```
data(eem_list)
eem_scale_ext(eem_list)
```

---

eem_setNA — *set parts of specific samples to NA and optionally interpolate these parts*

Description

set parts of specific samples to NA and optionally interpolate these parts

Usage

```
eem_setNA(eem_list, sample = NULL, em = NULL, ex = NULL,
  interpolate = TRUE, ...)
```

Arguments

`eem_list`	EEMs as eemlist
`sample`	optional, names or indices of samples to process
`em`	optional, emission wavelengths to set NA
`ex`	optional, excitation wavelengths to set NA
`interpolate`	FALSE, 1 or 2, interpolate NAs or not, 2 different methods, see `eem_interp`
`...`	arguments passed on to `eem_interp`

Details

Samples and wavelengths are optional and if not set all of them are considered in setting data to NA. Wavelengths can be set as vectors containing more than the wavelengths present in the data. E.g. 230:250 removes all wavelengths between 230 and 250 if present. Data is best interpolated if it does not reach data boundaries. Please check the results otherwise as in some cases the interpolation might not produce meaningful data.

Value

eemlist

Examples

```
data(eem_list)
eem <- eem_list[1:9]
class(eem) <- "eemlist"

ggeem(eem)

eem_list2 <- eem_setNA(eem,ex=200:280,em=500:600, interpolate=FALSE)
ggeem(eem_list2)
```

---

eem_smooth — *Smooth fluorescence data by calculating rolling mean along excitation wavelengths.*

Description

Smooth fluorescence data by calculating rolling mean along excitation wavelengths.

Usage

```
eem_smooth(data, n = 4)
```

Arguments

data	fluorescence data of class eemlist
n	width of rolling mean window in nm

Value

eemlist with smoothed data

Examples

```
data(eem_list)

eem_list <- eem_smooth(eem_list,n=4)
```

---

eem_spectral_cor — *Multiply EEMs with spectral correction vectors (Emission and Excitation)*

Description

Multiply EEMs with spectral correction vectors (Emission and Excitation)

Usage

```
eem_spectral_cor(eem_list, Excor, Emcor)
```

Arguments

eem_list	eemlist
Excor	data frame, first column wavelengths, second column excitation correction
Emcor	data frame, first column wavelengths, second column emission correction

Value

eemlist

Examples

```
eems <- system.file("extdata/EEMs",package="staRdom")
eem_list <- eem_read(eems, recursive = TRUE, import_function = eem_csv)

excorfile <- system.file("extdata/CorrectionFiles/xc06se06n.csv",package="staRdom")
Excor <- data.table::fread(excorfile)
emcorfile <- system.file("extdata/CorrectionFiles/mcorrs_4nm.csv",package="staRdom")
Emcor <- data.table::fread(emcorfile)

# adjust range of EEMs to cover correction vectors
eem_list <- eem_range(eem_list,ex = range(Excor[,1]), em = range(Emcor[,1]))

eem_list_sc <- eem_spectral_cor(eem_list,Excor,Emcor)
```

---

ggeem — *EEM spectra plotted with ggplot2*

Description

Plots from EEM spectra of class ggplot. In case you work with a larger number of EEMs and want to show then in several plots, you can use eem_overview_plot.

Usage

```
ggeem(data, fill_max = FALSE, ...)

## Default S3 method:
ggeem(data, fill_max = FALSE, ...)

## S3 method for class 'eemlist'
ggeem(data, fill_max = FALSE, eemlist_order = TRUE,
  ...)

## S3 method for class 'eem'
ggeem(data, fill_max = FALSE, ...)

## S3 method for class 'parafac'
ggeem(data, fill_max = FALSE, ...)

## S3 method for class 'data.frame'
ggeem(data, fill_max = FALSE, redneg = FALSE,
  contour = FALSE, interpolate = FALSE, ...)
```

Arguments

`data`	eem, eemlist, parafac or data.frame. The details are given under 'Details'.
`fill_max`	sets the maximum fluorescence value for the colour scale. This is mainly used by other functions, and makes different plots visually comparable.
`...`	parameters passed on to `ggplot`.
`eemlist_order`	logical, in case of an eemlist, the order of samples in the plot is the same as in the eemlist, alphabetically otherwise
`redneg`	logical, whether negative values should be coloured discreet.
`contour`	logical, whether contours should be plotted (default FALSE), see `geom_contour`
`interpolate`	logical, whether fluorescence should be interpolated, see `geom_raster`

Details

The data can be of different sources: eem: a single EEM spectrum is plotted eemlist: all spectra of the samples are plotted, arranged in a grid data.frame: a data.frame containing EEM data. Can be created by e.g. `as.data.frame.eem` parafac: a PARAFAC model, the components are plotted then.

Using redneg you can give negative values a reddish colour. This can help identifying these parts in samples or components. Negative values are physically not possible and can only be the result of measuring errors, model deviations and problems with interpolated values.

Interpolation (interpolate = TRUE) leeds to smoother plots. The default is FALSE because it might cover small scale inconsistencies.

Contours (contour = TRUE)can be added to the EEM plots.

A colour palette can be specified using the argument colpal.

Plotting distinct samples can be done using `eem_extract`. Please see example.

Value

a ggplot object

Examples

```
## plotting two distinct samples
data(eem_list)
eem_names(eem_list)
eem <- eem_extract(eem_list,c("^d667sf$", "^d661sf$"),keep=TRUE)
ggeem(eem)
ggeem(eem, interpolate = TRUE)
ggeem(eem, contour = TRUE)
```

list_join — *Full join of a list of data frames.*

Description

Full join of a list of data frames.

Usage

```
list_join(df_list, by)
```

Arguments

df_list	list of data frames to by joined
by	character vector containing information how to join data frames. Format to be according to by in full_join. Each data frame has to contain the column(s) used for joining.

Value

The joint data frame.

See Also

full_join

Examples

```
a <- data.frame(what=letters[1:5],a=c(1:5))
b <- data.frame(what=letters[1:5],b=c(7:11))
c <- data.frame(what=letters[1:5],c=c(20:24))

df_list <- list(a,b,c)

list_join(df_list,by="what")
```

maxlines — *Extract data from emission and excitation wavelengths of the components of a PARAFAC model (scaled B- and C-modes)*

Description

Data for each wavelengths is returned. For each component the lines intersecting at the component maxima are returned.

Usage

```
maxlines(pfmodel)
```

Arguments

pfmodel	object of class parafac

Value

data frame

Examples

```
data(pf_models)

ml <- maxlines(pf4[[1]])
```

---

norm2A — *Compensate for normalisation in C-modes*

Description

Factors used for normalisation are saved separately in the PARAFAC models. With this function, the normalisation factors are combined with the A-modes of the model and removed as a separate vector. This means former normalisation is accounted for in the amount of each component in each sample. If no normalisation was done, the original model is returned without warning.

Usage

```
norm2A(pfmodel)
```

Arguments

pfmodel	object of class parafac

Value

object of class parafac

Examples

```
data(pf_models)

pf4[[1]] <- norm2A(pf4[[1]])
```

norm_array — *Normalise 3-dimensional array in first and second dimension*

Description

Normalise 3-dimensional array in first and second dimension

Usage

```
norm_array(eem_array)
```

Arguments

`eem_array`	3-dimensional array

Value

array

Examples

```
data(eem_list)

a <- eem2array(eem_list)
an <- norm_array(a)
```

parafac_conv — *Calculate a PARAFAC model similar to and using parafac.*

Description

Please refer to parafac for input parameters and details. This wrapper function ensures ‘nstart‘ converging models are calculated. On the contrary, parafac calculates ‘nstart‘ models regardless if they are converging.

Usage

```
parafac_conv(X, nstart, verbose = FALSE, output = c("best", "all"),
  cl = NULL, ...)
```

Arguments

`X`	array
`nstart`	number of converging models to calculate
`verbose`	logical, whether more information is supplied
`output`	Output the best solution (default) or output all nstart solutions.
`cl`	cluster to be used for parallel processing
`...`	arguments passed on to parafac

Value

either a parafac model or a list of parafac models

See Also

`parafac`

Examples

```
# sorry, no example provided yet
```

`pf1` *PARAFAC model, see vignette, unconstrained*

Description

PARAFAC model, see vignette, unconstrained

Usage

```
pf1
```

Format

list of parafacs

`pf1n` *PARAFAC model, see vignette, non-negative constraints*

Description

PARAFAC model, see vignette, non-negative constraints

Usage

```
pf1n
```

Format

list of parafacs

`pf2` *PARAFAC model, see vignette, non-negative constraints, normalised*

Description

PARAFAC model, see vignette, non-negative constraints, normalised

Usage

```
pf2
```

Format

list of parafacs

`pf3` *PARAFAC model, see vignette, non-negative constraints, normalised, outliers removed*

Description

PARAFAC model, see vignette, non-negative constraints, normalised, outliers removed

Usage

```
pf3
```

Format

list of parafacs

`pf4` *PARAFAC model, see vignette, non-negative constraints, normalised, outliers removed, high accuarcy*

Description

PARAFAC model, see vignette, non-negative constraints, normalised, outliers removed, high accuarcy

Usage

```
pf4
```

Format

list of parafacs

sh — *result from PARAFAC split-half analysis, periodic data split*

Description

result from PARAFAC split-half analysis, periodic data split

Usage

```
sh
```

Format

list of parafacs

splithalf — *Running a Split-Half analysis on a PARAFAC model*

Description

The samples are split into four subsamples: A,B,C,D. Subsamples are then combined and compared: AB vs. CD, AC vs. BD, AD vs. BC. The results show graphs from the components of each of the 6 models.

Usage

```
splithalf(eem_list, comps, splits = NA, rand = FALSE,
  normalise = TRUE, nstart = 20,
  cores = parallel::detectCores(logical = FALSE), maxit = 2500,
  ctol = 10^(-7), rescale = TRUE, verbose = FALSE, ...)
```

Arguments

`eem_list`	eemlist containing sample data
`comps`	number of desired components
`splits`	optional, list of 4 numerical vectors containing the sample numbers for A,B,C and D sample subsets
`rand`	logical, splits are randomised
`normalise`	state whether EEM data should be normalised in advance
`nstart`	number of random starts
`cores`	number of parallel calculations (e.g. number of physical cores in CPU)
`maxit`	maximum iterations for PARAFAC algorithm
`ctol`	Convergence tolerance (R^2 change)
`rescale`	rescale splithalf models to Fmax, see `eempf_rescaleBC`
`verbose`	states whether you want additional information during calculation
`...`	additional parameters that are passed on to `parafac`

Details

Split data sets can be split suboptimal and cause low TCCs. Therefore, subsamples are recombined in 3 different ways and a TCC close to 1 in only one split combination per component is already a positive result. Check the split sets to check for sample independency.

Value

data frame containing components of the splithalf models

See Also

`splithalf_plot`, `splithalf_tcc`

Examples

```
data(eem_list)

splithalf <- splithalf(eem_list, comps = 6)
splithalf_plot(splithalf)
```

`splithalf_plot` *Plot results from a splithalf analysis*

Description

Graphs of all components of all models are plotted to be compared.

Usage

```
splithalf_plot(fits)
```

Arguments

`fits`	list of components data

Value

ggplot

See Also

`splithalf`

Examples

```
data(sh)

splithalf_plot(sh)
```

splithalf_splits — *Extracting a list of sample names in each subsample from a splithalf analysis*

Description

Extracting a list of sample names in each subsample from a splithalf analysis

Usage

```
splithalf_splits(fits)
```

Arguments

`fits`	list of parafac models (from a splithalf analysis)

Value

data frame containing TCC values

Examples

```
data(sh)
splithalf_splits(sh)
```

splithalf_tcc — *Extracting TCC values from a splithalf analysis*

Description

Extracting TCC values from a splithalf analysis

Usage

```
splithalf_tcc(fits)
```

Arguments

`fits`	list of parafac models (from a splithalf analysis)

Value

data frame containing TCC values

Examples

```
data(sh)

splithalf_tcc(sh)
```

`ssc` *Calculate the shift-and shape-sensitive congruence (SSC) between two matrices*

Description

Please see details in: U.J. Wünsch, R. Bro, C.A. Stedmon, P. Wenig, K.R. Murphy, Emerging patterns in the global distribution of dissolved matter fluorescence, Anal. Methods, 11 (2019), pp. 888-893

Usage

```
ssc(mat1, mat2, tcc = FALSE)
```

Arguments

`mat1`	matrix
`mat2`	matrix
`tcc`	if set TRUE, TCC is returned instead

Value

table containing pairwise SCC of matrices columns

Examples

```
pf_models <- pf3
mat1 <- pf_models[[1]][[2]]
mat2 <- pf_models[[2]][[2]]

## calculate SSC
ssc(mat1,mat2)

## calculate TCC
ssc(mat1,mat2, tcc = TRUE)
```

`ssc_max` *Calculate the combination of components giving the maximum of geometric mean of TCCs*

Description

Calculate the combination of components giving the maximum of geometric mean of TCCs

Usage

```
ssc_max(mat)
```

Arguments

`mat`	matrix

Value

vector with TCCs having the highest possible geometric mean

Examples

```
mat <- matrix(c(7,2,13,6,0,7,1,5,5), nrow = 3)
mat

sscs <- ssc_max(mat)
sscs

# order of components:
attr(sscs,"order")
```

`tcc` *Caluclate Tucker's Congruence Coefficient of PARAFAC components*

Description

Componets must be passed as modes, see `maxlines`

Usage

```
tcc(maxl_table, na.action = "na.omit")
```

Arguments

`maxl_table`	data frame containing the peak lines of components
`na.action`	if "na.omit" NA are deleted from prior the test

Value

data.frame containing the TCCs

Examples

```
data(pf_models)

ml <- maxlines(pf4[[1]])

tcc(ml)
```

`tcc_find_pairs` *Reorders components of different PARAFAC models according to best fit (TCC)*

Description

When running a splithalf analysis similar components are not necessarily on the same position. This function looks for best fits with Tucker's Congruence Coefficients and returns a list of models with reordered components.

Usage

```
tcc_find_pairs(fits)
```

Arguments

`fits`	list of parafac models

Value

list of parafac models

See Also

splithalf

Examples

```
data(eem_list)

# function currently only used from within splithalf
splithalf(eem_list,6,nstart=2)
```

Index