Package 'albatross'

October 2, 2020

Type Package

Title PARAFAC Analysis of Fluorescence Excitation-Emission Matrices

Version 0.1-2

Date 2020-10-02

Depends R (>= 2.10)

Imports multiway (>= 1.0-4), pracma, lattice

Description Perform parallel factor analysis (PARAFAC: Hitchcock, 1927) <doi:10.1002/sapm192761164> on fluorescence excitation-emission matrices (FEEMs): handle scattering signal and inner filter effect, scale the dataset, fit the model; perform split-half validation or jack-knifing. A modified approach called ``randomised split-half'' is also available. The package has a low dependency footprint (only two direct dependencies not in core or recommended; four total non-core/recommended dependencies) and has been tested on a wide range of R versions (including R as old as 3.3.3 from Debian Stretch).

License GPL (>= 3)

NeedsCompilation no

Author Ivan Krylov [aut, cre],
Timur Labutin [ths]

Maintainer Ivan Krylov <ikrylov@laser.chem.msu.ru>

Repository CRAN

Date/Publication 2020-10-02 12:42:05 UTC

R topics documented:

`albatross-package` *PARAFAC Analysis of Fluorescence Excitation-Emission Matrices*

Description

Day after day, day after day,
We stuck, nor breath nor motion;
As idle as a painted ship
Upon a painted ocean.

Water, water, every where,
And all the boards did shrink;
Water, water, every where,
Nor any drop to drink.

– Samuel Taylor Coleridge, *The Rime of the Ancient Mariner*

Perform parallel factor analysis (PARAFAC: Hitchcock, 1927) <doi:10.1002/sapm192761164> on fluorescence excitation-emission matrices (FEEMs): handle scattering signal and inner filter effect, scale the dataset, fit the model; perform split-half validation or jack-knifing. A modified approach called "randomised split-half" is also available. The package has a low dependency footprint (only two direct dependencies not in core or recommended; four total non-core/recommended dependencies) and has been tested on a wide range of R versions (including R as old as 3.3.3 from Debian Stretch).

Details

Index of help topics:

```
[.feem                  Extract or replace parts of FEEM objects
[.feemcube              Extract or replace parts of FEEM cubes
albatross-package       PARAFAC Analysis of Fluorescence
                        Excitation-Emission Matrices
as.data.frame.feem      Transform a FEEM object into a data.frame
as.list.feemcube        Transform a FEEM cube object into a list of
                        FEEM objects
feem                    Create a fluorescence excitation-emission
```

	matrix object
feemcube	Build a data cube of FEEMs
feemife	Absorbance-based inner filter effect correction
feemjackknife	Jack-knife outlier detection in PARAFAC models
feemparafac	Compute PARAFAC on a FEEM cube object
feems	Fluorescence excitation-emission matrices and absorbance spectra
feemscale	Rescale FEEM spectra to a given norm and remember the scale factor
feemscatter	Handle scattering signal in FEEMs
feemsplithalf	Split-half analysis of PARAFAC models
fitted.feemparafac	Extract fitted PARAFAC values or residuals
marine.colours	Marine colours
plot.feem	Plot a FEEM object

Author(s)

Timur Labutin

Maintainer: Ivan Krylov

References

K.R. Murphy, C.A. Stedmon, D. Graeber, R. Bro, Fluorescence spectroscopy and multi-way techniques. PARAFAC, Analytical Methods. 5 (2013) 6557. doi: 10.1039/c3ay41160e

M. Pucher, U. Wünsch, G. Weigelhofer, K. Murphy, T. Hein, D. Graeber, staRdom: Versatile Software for Analyzing Spectroscopic Data of Dissolved Organic Matter in R, Water. 11 (2019) 2366. doi: 10.3390/w11112366

J. Cleese, T. Jones, Albatross: Flavours of different sea birds, Journal of Flying Circus. 1.13 (1970) 7:05-7:45.

See Also

feem, feemcube, feemscatter, feemife, feemscale, feemparafac, feemsplithalf, feemjackknife.

Examples

```
plot(x <- feem(matrix(1:42, 7), 400:406, 350:355))

data(feems)

dataset <- feemcube(feems, FALSE)[1:30*6, 1:9*6,]
dataset <- feemscatter(dataset, rep(24, 4), 'pchip')
dataset <- feemife(dataset, absorp)
plot(dataset <- feemscale(dataset, na.rm = TRUE))


  # takes a long time
  (sh <- feemsplithalf(
    dataset, nfac = 2:5, const = rep('nonneg', 3), splits = 4)
  )
```

```
  plot(sh)
  jk <- feemjackknife(dataset, nfac = 3, const = rep('nonneg', 3))
  plot(jk)


pf <- feemparafac(dataset, nfac = 2, const = rep('nonneg', 3))
plot(pf)
```

`as.data.frame.feem` *Transform a FEEM object into a data.frame*

Description

Transform a FEEM object from its matrix form accompanied by vectors of wavelengths into a three-column form consisting of $(\lambda_{em}, \lambda_{ex}, I)$ tuples, which could be useful for export or plotting with **lattice** or **ggplot2**.

Usage

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

Arguments

`x`	A FEEM object, or a FEEM cube object.
`row.names`	Passed to `data.frame`. If default of `NULL` is used, `data.frame` will generate sequential integer `row.names`.
`optional`	This option is required for compatibility with `as.data.frame` generic, but is ignored, since column names are already syntactic and row names are generated by `data.frame` automatically by default.
`...`	Passed as-is to `data.frame`.

Details

Rows where intensity is `NA` are omitted from the output.

Value

A `data.frame` containing three numeric columns:

`emission`	Emission wavelength, nm
`excitation`	Excitation wavelentgh, nm
`intensity`	Fluorescence intensity at $(\lambda_{em}, \lambda_{ex})$
`sample`	For FEEM cube objects, the unique name of the sample possessing this tuple of values, a factor. If the original object didn't have any names, sequential integers are used instead. If the original object had non-unique names, sequence numbers are appended to them using `make.unique`.

See Also

feem.data.frame

Examples

```
z <- feem(matrix(1:42, nrow = 7), 1:7, 1:6)
head(as.data.frame(z))
```

`as.list.feemcube` *Transform a FEEM cube object into a list of FEEM objects*

Description

Return a list of FEEM objects comprising it. Used internally in `.feemcube` methods of the package generics and in `as.data.frame.feemcube`, but may be useful elsewhere.

Usage

```
  ## S3 method for class 'feemcube'
as.list(x, ...)
```

Arguments

`x`	A FEEM cube object.
`...`	No arguments besides those specified above are allowed.

Value

A named list of FEEM objects comprising `x`.

See Also

`as.list`; `feemcube` and its `list` constructor.

Examples

```
str(as.list(feemcube(array(1:60, 3:5), 1:3, 1:4)))
```

feem *Create a fluorescence excitation-emission matrix object*

Description

Functions to create fluorescence excitation-emission matrix objects from R matrices coupled with excitation and emission wavelengths or three-column `data.frames` containing $(\lambda_{em}, \lambda_{ex}, I)$ tuples.

Usage

```
  feem(x, ...)
  ## S3 method for class 'matrix'
feem(x, emission, excitation, scale = 1, ...)
  ## S3 method for class 'data.frame'
feem(
    x, scale = 1, emission = 'emission',
    excitation = 'excitation', intensity = 'intensity', ...
  )
```

Arguments

`x`	The matrix or three-column `data.frame` to convert into a FEEM object. If converting a matrix, its rows should correspond to different fluorescence emission wavelengths specified in `emission` argument; conversely, its columns should correspond to excitation wavelengths specified in `excitation` argument. If converting a `data.frame`, it should have exactly three columns, one for emission wavelengths, excitation wavelength, and intensity. The names of the columns are expected to be `'emission'`, `'excitation'`, and `'intensity'`, respectively, but can be overridden using namesake arguments.
`emission`	If conversing a matrix, this should be a vector of emission wavelengths, each wavelength corresponding to a row of the matrix. If converting a `data.frame`, this optional argument specifies the name of the column containing the emission wavelengths.
`excitation`	If conversing a matrix, this should be a vector of excitation wavelengths, each wavelength corresponding to a column of the matrix. If converting a `data.frame`, this optional argument specifies the name of the column containing the excitation wavelengths.
`intensity`	If converting a `data.frame`, this optional argument specifies the name of the column containing the fluorescence intensities.
`scale`	The scale value of a EEM is preserved through the analysis procedure to make it easy to undo the scaling after runnining PARAFAC. If the EEM has been pre-multiplied prior to creating the FEEM object, you can set the multiplier here.
`...`	Extra arguments besides those specified above are not allowed.

Value

A FEEM object is a matrix with the following attributes added:

`emission`	Fluorescence emission wavelengths corresponding to the rows of the matrix, nm.
`excitation`	Fluorescence excitation wavelengths corresponding to the columns of the matrix, nm.
`dimnames`	Dimension names, copies of information above. Used only for presentation purposes.
`scale`	Scale factor, preserved through the analysis, which may be used later to undo the scaling. Initially 1.

See Also

FEEM methods: `plot.feem`, `as.data.frame.feem`, `[.feem`, `feemife`, `feemscale`, `feemscatter`.

Examples

```
feem(matrix(1:40, ncol = 8), 1:5, 1:8)
feem(
  data.frame(x = 1:10, y = 21:30, z = 31:40),
  emission = 'x', excitation = 'y', intensity = 'z'
)
```

---

feemcube *Build a data cube of FEEMs*

Description

This function builds tagged 3-dimensional arrays of fluorescence excitation-emission spectra. Given a list of FEEM objects, it can determine the range of their wavelengths. Otherwise, the object is created from the supplied numeric array and vectors of wavelengths and sample names.

Usage

```
  feemcube(x, ...)
  ## S3 method for class 'list'
feemcube(x, all.wavelengths, ...)
  ## S3 method for class 'array'
feemcube(x, emission, excitation, scales, names = NULL, ...)
```

Arguments

x	A list of FEEM objects, possibly named. Alternatively, a numeric array.
all.wavelengths	Logical, a flag specifying whether to include wavelengths not present in *all* of the samples. If FALSE, only those wavelength present in all of the samples are included.
emission	Numeric vector of emission wavelengths. Should correspond to the first dimension of the array x.
excitation	Numeric vector of excitation wavelengths. Should correspond to the second dimension of the array x.
scales	Numeric vector of scale factors corresponding to the spectra in the array. Should correspond to the third dimension of the array x. If missing, assumed to be all 1.
names	Character vector of names of the samples. Should correspond to the third dimension of the array x.
...	Additional arguments besides those specified above are not allowed.

Details

feemcube.list can be used to build FEEM data cubes from lists of FEEM objects even if their wavelength grids do not exactly match. The missing wavelengths are set to NA.

Value

A FEEM data cube is a numeric three-dimensional array with the following attributes:

emission	Fluorescence emission wavelengths corresponding to the first dimension of the array, nm.
excitation	Fluorescence excitation wavelengths corresponding to the second dimension of the array, nm.
dimnames	Dimension names, copies of information above. Used only for presentation purposes.
scales	Scale factors of the samples, corresponding to the third dimension of the array. Assumed to be 1 missing if not specified by the user.

See Also

FEEM cube methods: [.feemcube, plot.feemcube, as.data.frame.feemcube, as.list.feemcube, feemife, feemscale, feemscatter.

Examples

```
feemcube(array(1:24, 4:2), 1:4, 1:3) # array form
feemcube(replicate(2, feem(matrix(1:6, 2), 1:2, 1:3), FALSE), TRUE) # list form
```

`feemife` *Absorbance-based inner filter effect correction*

Description

Use absorbance data to correct inner-filter effect in FEEM objects and collections of them.

Usage

```
  feemife(x, ...)
  ## S3 method for class 'feem'
feemife(x, absorbance, abs.path = 1, ...)
  ## S3 method for class 'feemcube'
feemife(x, absorbance, abs.path, ...)
  ## S3 method for class 'list'
feemife(x, absorbance, abs.path, ...)
```

Arguments

`x`	A FEEM object, a FEEM data cube, or a list of them.
`absorbance`	If `x` is a FEEM object: a two-column matrix-like object suitable for `xy.coords` containing the absorbance spectrum of the sample: the wavelengths in the first column and the unitless absorbance values in the second column. Otherwise, this could be a list of such objects or a multi-column matrix-like objects. If `x` contains names of the samples (is a named list or had `names` specified when calling `feemcube`), `absorbance` is a named list or has named columns, and all samples from `x` can be looked up in `absorbance`, results of this lookup are used. If name lookup fails but (given N-sample `x`) `absorbance` has exactly $N + 1$ columns or is an N-element list, absorbances are supposed to be present in the same order as the samples in `x`.
`abs.path`	If `x` is a FEEM object, a number specifying the length of the optical path used when measuring the absorbance, cm. Otherwise, a named vector containing the names from `x`, or a vector of exactly same length as the number of FEEMs in `x`: same lookup rules apply as for `absorbance` argument. If not set, assumed to be 1.
`...`	No parameters besides those described above are allowed.

Details

The formula used is:

$$I_{corr}(\lambda_{em}, \lambda_{ex}) = I_{orig}(\lambda_{em}, \lambda_{ex}) 10^{\frac{A(\lambda em) + A(\lambda ex)}{2L_{abs}}}$$

Value

An object of the same kind as `x`, with inner filter effect corrected.

References

J.R. Lakowicz, Principles of Fluorescence Spectroscopy, 3rd ed., Springer US, 2006. https://www.springer.com/la/book/9780387312781

D.N. Kothawala, K.R. Murphy, C.A. Stedmon, G.A. Weyhenmeyer, L.J. Tranvik, Inner filter correction of dissolved organic matter fluorescence: Correction of inner filter effects, Limnology and Oceanography: Methods. 11 (2013) 616-630. doi: 10.4319/lom.2013.11.616

Examples

```
data(feems)

dataset <- feemcube(feems, FALSE)
str(dataset)
str(absorp)
plot(feemife(dataset,absorp) / dataset)
```

`feemjackknife` *Jack-knife outlier detection in PARAFAC models*

Description

Perform leave-one-out fitting + validation of PARAFAC models on a given FEEM cube.

Usage

```
  feemjackknife(cube, ...)
  ## S3 method for class 'feemjackknife'
plot(
    x, kind = c('estimations', 'RIP', 'IMP'), ...
  )
```

Arguments

cube	A `feemcube` object.
x	An object returned by `feemjackknife`.
kind	`"estimations"` Plot the loadings from every leave-one-out model. `"RIP"` Produce a Resample Influence Plot, i.e. plot mean squared difference between loadings in overall and leave-one-out models against mean squared residuals in leave-one-out models. `"IMP"` Produce an Identity Match Plot, i.e. plot scores in leave-one-out models against scores in overall model.
...	**feemjackknife** Passed as-is to `feemparafac` and, eventually, to `parafac` **plot.feemjackknife** When `kind` is `"RIP"` or `"IMP"`, pass a `q` argument to specify the quantile of residual values (for RIP) or absolute score differences (IMP) above which sample names (or numbers) should be plotted. Remaining arguments are passed as-is to `xyplot`.

Details

The function takes each sample out of the dataset, fits a PARAFAC model without it, then fits the outstanding sample to the model with emission and excitation factors fixed.

The individual leave-one-out models are reordered according to best Tucker's congruence coefficient match and rescaled by minimising $||\mathbf{A}\,\mathrm{diag}(\mathbf{s}_\mathrm{A}) - \mathbf{A}^\mathrm{orig}||^2$ and $||\mathbf{B}\,\mathrm{diag}(\mathbf{s}_\mathrm{B}) - \mathbf{B}^\mathrm{orig}||^2$ over $\mathbf{s}_\mathrm{A}$ and $\mathbf{s}_\mathrm{B}$, subject to $\mathrm{diag}(\mathbf{s}_\mathrm{A})\,\mathrm{diag}(\mathbf{s}_\mathrm{B})\,\mathrm{diag}(\mathbf{s}_\mathrm{C}) = \mathbf{I}$, to make them comparable.

Once the models are fitted, resample influence plots and identity match plots can be produced from resulting data to detect outliers.

It is recommended to fully name the parameters to be passed to `feemparafac` to avoid problems.

`plot.feemjackknife` provides sane defaults for `xyplot` parameters `xlab`, `ylab`, `scales`, `as.table`, but they can be overridden.

Value

feemjackknife An list of class `feemjackknife` containing the following entries:

- **overall** Result of fitting the overall cube with `feemparafac`.
- **leaveone** A list of length `dim(cube)[3]` containing the reduced dataset components. Every `feemparafac` object in the list has an additional `Chat` attribute containing the result of fitting the excluded spectrum back to the loadings of the reduced model.

plot.feemjackknife A **lattice** plot object. Its `print` or `plot` method will draw the plot on an appropriate plotting device.

References

J. Riu, R. Bro, Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models, Chemometrics and Intelligent Laboratory Systems. 65 (2003) 35-49. doi: 10.1016/S01697439(02)000904

See Also

`feemparafac`

Examples

```
data(feems)
cube <- feemscale(
  feemscatter(feemcube(feems, FALSE), rep(24, 4))[1:30*6, 1:9*6,],
  na.rm = TRUE
)
# takes a long time
jk <- feemjackknife(cube, nfac = 3, const = rep('nonneg', 3))
plot(jk)
plot(jk, 'IMP')
plot(jk, 'RIP')
```

`feemparafac` *Compute PARAFAC on a FEEM cube object*

Description

This function forwards its arguments to `parafac` from the **multiway** package, optionally rescales the result and attaches a few attributes.

Usage

```
  feemparafac(X, ..., rescale = 3, retries = 10)
  ## S3 method for class 'feemparafac'
plot(x, type = c("image", "lines"), ...)
```

Arguments

`X`	A FEEM cube object. The per-sample factors will be multiplied by `attr(X,'scales')` stored in it.
`...`	**feemparafac** Passed as-is to `parafac`. **plot.feemparafac** Passed as-is to **lattice** functions `levelplot` and `xyplot`.
`rescale`	Rescale the resulting factors to leave all the variance in the given mode: emission, excitation, or sample (default). Set to `NA` to disable.
`retries`	Retry for given number of tries until `parafac` returns a successfully fitted model or stops due to the iteration number limit. Raise a fatal error if all tries were unsuccessful.
`x`	An object returned by `feemparafac`.
`type`	Given a fitted PARAFAC model:

$$X_{ijk} = \sum_r A_{ir} B_{jr} C_{kr}$$

With $\mathbf{A}$ corresponding to fluorescence emission spectra, $\mathbf{B}$ corresponding to fluorescence excitation spectra, and $\mathbf{C}$ corresponding to the scores of the components in different samples, the following plots can be produced:

`"image"` Plot the factors ("loadings") as a series of false-colour images of outer products $\mathbf{A}_r \otimes \mathbf{B}_r$.

`"lines"` Plot the factors $\mathbf{A}_r$ and $\mathbf{B}_r$ as functions of wavelengths, with each pair of factors on a different panel.

More plot kinds may be added in the future.

Details

The function tries hard to guarantee the convergence flag to be `0` (normal convergence) or `1` (iteration number limit reached), but never `2` (a problem with the constraints). A fatal error is raised if repeated runs of `parafac` do not return a (semi-)successfully fitted model.

The `output` option is fixed to `"best"` value. Obtaining a list of alternative solutions can therefore be achieved by running:

`replicate(`*n*`,feemparafac(...,nstart = 1))`

`plot.feemparafac` provides sane defaults for **lattice** options such as `xlab`, `ylab`, `as.table`, `auto.key`, `type`, `cuts`, `col.regions`, but they can be overridden.

Value

`feemparafac` An object of classes `feemparafac` and `parafac` with the `cube` attribute added containing a reference to the original FEEM cube object.

`plot.feemparafac`
A **lattice** plot object. Its `print` or `plot` method will draw the plot on an appropriate plotting device.

References

R. Bro, PARAFAC. Tutorial and applications, Chemometrics and Intelligent Laboratory Systems. 38 (1997) 149-171. doi: 10.1016/S01697439(97)000324

See Also

`parafac` for the `parafac` class structure; `fitted.feemparafac`, `residuals.feemparafac`, for methods specific to values returned from this function.

`levelplot`, `xyplot`.

Examples

```
data(feems)
cube <- feemscale(
  feemscatter(
    feemcube(feems, FALSE)[(1:45)*4,(1:13)*4,],
    rep(24, 4)), na.rm = TRUE
)
factors <- feemparafac(cube, nfac = 2, const = rep('nonneg', 3))
plot(factors, 'image')
plot(factors, 'line')
```

feems *Fluorescence excitation-emission matrices and absorbance spectra*

Description

This dataset contains:

- Twelve FEEMs with anti-Stokes zone missing and scattering signal not handled, captured at slightly different wavelength grids and with some points missing.
- Twelve absorbance spectra for purposes of IFE correction.

Usage

```
data("feems")
```

Format

feems A named list of 12 feem objects containing fluorescence data measured at different wavelength grids (excitation wavelengths between 230 nm and 500 or 550 nm; emission wavelengths between 240 nm and 600 or 650 nm). Intensity at $\lambda_{em} < \lambda_{ex} + 10$ nm is not measured.

absorp A named list of 12 2-column matrices containing absorbance spectra measured between 230 and 650 nm in 1 cm cells.

Examples

```
data(feems)
plot(feems$a)
plot(absorp$a)
```

feemscale *Rescale FEEM spectra to a given norm and remember the scale factor*

Description

Given a norm function (typically, standard deviation), scale the intensities in FEEM objects to it and optionally remember the scale factor.

Usage

```
  feemscale(x, ...)
  ## S3 method for class 'feem'
feemscale(x, norm = sd, remember = TRUE, ...)
  ## S3 method for class 'feemcube'
feemscale(x, ...)
  ## S3 method for class 'list'
feemscale(x, ...)
```

Arguments

x	A FEEM object, a FEEM cube object, or a list of anything compatible with feemscale generic.
norm	A function taking a numeric matrix and returning its norm. Typically, sd or sumsq.
remember	Whether to remember the scale factor. If FALSE, the scale factor in the returned object is unchanged.
...	Passed as-is to feemscale and, eventually, to feemscale.feem. Also passed as-is to the norm function, which is useful for passing na.rm = TRUE to functions like sd or sumsq.

Value

feemscale.feem: a FEEM object with intensities divided by scale factor (norm(x)) and its scale attribute multiplied by the scale factor.

feemscale.feemcube: a FEEM cube built from FEEM objects scaled as described above.

feemscale.list: a list consisting of results of feemscale generic applied to its elements.

See Also

feem

Examples

```
feemscale(feem(matrix(1:42, 6), 1:6, 1:7))
```

---

feemscatter *Handle scattering signal in FEEMs*

Description

Remove or interpolate scattering signal in individual FEEM objects, FEEM cube objects, or lists of them.

Usage

```
  feemscatter(x, ...)
  ## S3 method for class 'list'
feemscatter(x, ...)
  ## S3 method for class 'feemcube'
feemscatter(x, ...)
  ## S3 method for class 'feem'
feemscatter(
    x, widths, method = c("omit", "pchip", "loess"),
    add.zeroes = 30, Raman.shift = 3400, ...
  )
```

Arguments

`x`	An individual FEEM object, FEEM cube object, or a list of them, to handle the scattering signal in.
`widths`	A numeric vector of length 4 containing the widths (in nm) of the scattering signal, in the following order: 1. Rayleigh scattering 2. Raman scattering 3. Rayleigh scattering, 2λ 4. Raman scattering, 2λ Set a width to 0 if you don't want to handle this particular kind of scattering signal.
`method`	A string choosing *how* to handle the scattering signal: `"omit"` Replace it with `NA` `"pchip"` Interpolate it line-by-line using piecewise cubic Hermitean polynomials (`pchip`). `"loess"` Interpolate it by fitting a locally weighted polynomial surface (`loess`). In this case the remaining parameters are passed verbatim to `loess`, which may be used to set parameters such as `span`.
`add.zeroes`	Set intensities at $\lambda_{em} < \lambda_{ex} -$ `add.zeroes` nm to 0 unless they have been measured. Set to `NA` to disable this behaviour.
`Raman.shift`	Raman shift of the scattering signal of water, cm^{-1}.
`...`	Passed verbatim from `feemscatter` generics to `feemscatter.feem`. If `"loess"` method is selected, remaining options are passed to `loess` (the `span` parameter is of particular interest there).

Details

The `"pchip"` method is implemented as described in the article by Bahram et al., 2006 (see the reference below): each emission spectrum at different excitation wavelengths is considered one by one. The last excitation spectrum (highest emission wavelength) is interpolated first to handle the corner case of extrapolating 2λ scattering near the high end of the emission wavelength range. Zero-valued points are inserted at the lowest emission wavelength (unless the data is already present) to prevent the extrapolation from blowing up.

The `"loess"` method feeds the whole FEEM except the area to be interpolated to `loess`, then asks it to predict the remaining part of the spectum.

Value

An object of the same kind (FEEM object / FEEM cube / list of them) with scattering signal handled as requested.

References

M. Bahram, R. Bro, C. Stedmon, A. Afkhami, Handling of Rayleigh and Raman scatter for PARAFAC modeling of fluorescence data using interpolation, Journal of Chemometrics. 20 (2006) 99-105. doi: 10.1002/cem.978

See Also

feem, feemcube

Examples

```
data(feems)
plot(x <- feemscatter(
  feems[[1]], widths = c(25, 25, 20, 20), method = "loess", Raman.shift = 3500,
  span = .03, control = loess.control(trace.hat = 'approximate')
))
```

---

feemsplithalf *Split-half analysis of PARAFAC models*

Description

This function validates PARAFAC with different numbers of components by means of splitting the data cube in halves, fitting PARAFAC to them and comparing the results [1].

Usage

```
  feemsplithalf(cube, nfac, splits, random, ...)
  ## S3 method for class 'feemsplithalf'
plot(x, kind = c("tcc", "factors"), ...)
  ## S3 method for class 'feemsplithalf'
print(x, ...)
```

Arguments

cube	A feemcube object.
nfac	An integer vector of numbers of factors to check.
splits	Number of parts to split the data cube into. Must be even. After splitting, all ways to recombine the parts into non-intersecting halves are enumerated [2], the halves are subjected to PARAFAC decomposition and compared against each other. The number of PARAFAC models fitted is $\binom{\texttt{splits}}{\texttt{splits}/2}$. Mutually incompatible with random parameter.
random	Number of times to shuffle the dataset, split into non-intersecting halves, fit a PARAFAC model to each of the halves and compare halves against each other. The number of PARAFAC models fitted is $2 \cdot \texttt{random}$. Mutually incompatible with splits parameter.
x	An object returned by feemsplithalf.
kind	"tcc" Display statistics of between-half TCCs for different numbers of components. The smallest TCC is chosen between emission- and excitation-mode values, but otherwise they are not aggregated. Components with the same number have the same colour.

`"factors"` Plot the resulting factors themselves on panels per each number of components and each mode (emission or excitation). Components with the same number have the same colour.

... **feemsplithalf** Remaining options are passed to `feemparafac` and, eventually, to `parafac`.

plot.feemsplithalf Passed as-is to `xyplot`.

print.feemsplithalf No options are allowed.

Details

Pass either `splits` or `random` parameter, but not both, as they are mutually exclusive.

As the models are fitted, they are compared to the first model of the same number of factors (Tucker's congruence coefficient is calculated using `congru` for emission and excitation mode factors, then the smallest value of the two is chosen for the purposes of matching). The models are first reordered according to the best match by TCC value, then rescaled [3] by minimising $||\mathbf{A}\,\mathrm{diag}(\mathbf{s}_\mathrm{A}) - \mathbf{A}^\mathrm{orig}||^2$ and $||\mathbf{B}\,\mathrm{diag}(\mathbf{s}_\mathrm{B}) - \mathbf{B}^\mathrm{orig}||^2$ over $\mathbf{s}_\mathrm{A}$ and $\mathbf{s}_\mathrm{B}$, subject to $\mathrm{diag}(\mathbf{s}_\mathrm{A})\,\mathrm{diag}(\mathbf{s}_\mathrm{B})\,\mathrm{diag}(\mathbf{s}_\mathrm{C}) = \mathbf{I}$, to make them comparable.

`plot.feemsplithalf` provides sane defaults for `xyplot` parameters `xlab`, `ylab`, `as.table`, but they can be overridden.

Value

feemsplithalf, print.feemsplithalf An object of class `feemsplithalf`, containing named fields:

- **factors** A `list` of `feemparafac` objects containing the factors of the halves. The list has dimensions, the first one corresponding to the halves (always 2), the second to different numbers of factors (as many as in `nfac`) and the third to different groupings of the samples (depends on `splits` or `random`).
- **tcc** A named list containing arrays of Tucker's congruence coefficients between the halves. Each entry in the list corresponds to an element in the `nfac` argument. The dimensions of each array in the list correspond to, in order: the factors (1 to *nfac*), the modes (emission or excitation) and the groupings of the samples (depending on `splits` or `random`).
- **nfac** A copy of `nfac` argument.

plot.feemsplithalf A **lattice** plot object. Its `print` or `plot` method will draw the plot on an appropriate plotting device.

References

1. W.S. DeSarbo, An Application of PARAFAC to a Small Sample Problem, Demonstrating Preprocessing, Orthogonality Constraints, and Split-Half Diagnostic Techniques (Appendix), Social Science Research Network, Rochester, NY, 1984. https://papers.ssrn.com/abstract=2783446
2. K.R. Murphy, C.A. Stedmon, D. Graeber, R. Bro, Fluorescence spectroscopy and multi-way techniques. PARAFAC, Analytical Methods. 5 (2013) 6557. doi: 10.1039/c3ay41160e
3. J. Riu, R. Bro, Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models, Chemometrics and Intelligent Laboratory Systems. 65 (2003) 35-49. doi: 10.1016/S01697439(02)000904

See Also

feemparafac, parafac, congru, feemcube.

Examples

```
data(feems)
cube <- feemscale(
  feemscatter(feemcube(feems, FALSE), rep(24, 4))[1:30*6, 1:9*6,],
  na.rm = TRUE
)
(sh <- feemsplithalf( # takes a long time
  cube, 2:4, splits = 4, # 4 splits => S4C6T3
  # the rest is passed to multiway::parafac
  const = rep('nonneg', 3) # setting ctol and maxit is recommended
))
plot(sh)
plot(sh, 'factors')
```

---

fitted.feemparafac *Extract fitted PARAFAC values or residuals*

Description

`fitted` calculates an approximation of a FEEM cube fitted by PARAFAC.

`residuals` returns the difference between `fitted` and the original data as a FEEM cube.

Usage

```
  ## S3 method for class 'feemparafac'
fitted(object, ...)
  ## S3 method for class 'feemparafac'
residuals(object, ...)
```

Arguments

`object`	An object returned by feemparafac.
`...`	No arguments besides those described above are allowed.

Details

The output of fitted.parafac from **multiway** package is rescaled back according to saved scales (typically those from feemscale, e.g. sd norms of each spectrum) from the original cube in order to be comparable with it.

The output of `residuals` is $\mathbf{X} - \hat{\mathbf{X}}$.

Value

A FEEM cube object.

See Also

feemcube, fitted.parafac, resid.

Examples

```
data(feems)
cube <- feemscale(
  feemscatter(
    feemcube(feems, FALSE)[1:36*5, 1:11*5,],
    rep(24, 4)), na.rm = TRUE
)
factors <- feemparafac(cube, nfac = 2, const = rep('nonneg', 3))
# calls plot.feemcube for estimated spectra
plot(fitted(factors))
plot(resid(factors))
```

---

marine.colours *Marine colours*

Description

Create a perceptually contigous palette of R "#RRGGBB" colours, using hues typically associated with natural waters.

Usage

```
marine.colours(
  n, chroma = 0.65, luminance = c(0.35, 1),
  alpha = 1, gamma = 1, fixup = TRUE
)
```

Arguments

n	Number of colours to return.
chroma	Specifies the chroma (how saturated should the colours be) for the palette, a real number between 0 and 1. May also be a two-element vector, in which case the chroma is changed smoothly from start to finish of the resulting palette.
luminance	Specifies the luminance (how bright should the colours be) of the colours constituting the palette. Typically, a two-element vector of real numbers between 0 and 1 to indicate smooth change along the palette, but can also be a fixed number.
alpha	Specifies the transparency of the colours of the palette. As above, can be a fixed number or a two-element vector in the range $[0, 1]$. Typically, fully opaque (alpha=1) colours are used.

gamma — Provides the power coefficient for the hue/chroma/luminance growth formulae. May be useful when it is needed to sacrifice the perceptual linearity of the palette to provide more constrast for smaller or bigger values on the plot. The gamma-corrected values are obtained by computing x^γ, x in $[0; 1]$, then scaling the result linearly to the required range. Typically, linear growth (`gamma = 1`) is preferred.

fixup — Whether to correct the palette if the resulting colours happen to fall outside the valid RGB range (passed as-is to `hcl`). Unrepresentable colours are returned as `NA`s, but fixing the palette may make it less perceptually uniform.

Value

A character vector of length `n` containing colour specifications for use with R graphics functions.

References

Insired by cmocean palette called "haline" (https://matplotlib.org/cmocean/#haline), but using R's implementation of polar CIE-LUV colour space instead of CAM02-UCS.

See Also

`hcl` for the colour space used, CUBEHELIX (http://www.mrao.cam.ac.uk/~dag/CUBEHELIX/) for a similar technique using BT.601 luminance coefficients and RGB colour space.

Examples

```
image(volcano, col = marine.colours(256))
```

---

plot.feem *Plot a FEEM object*

Description

A thin wrapper around **lattice**::`levelplot`, which uses it to plot a 2D fluorescence intensity surface.

Usage

```
## S3 method for class 'feem'
plot(
  x,
  xlab = quote(lambda[em] * ", nm"), ylab = quote(lambda[ex] * ", nm"),
  cuts = 128, col.regions = marine.colours(256), ...
)
## S3 method for class 'feemcube'
plot(
  x,
  xlab = quote(lambda[em] * ", nm"), ylab = quote(lambda[ex] * ", nm"),
  cuts = 128, col.regions = marine.colours(256), as.table = TRUE, ...
)
```

Arguments

x	An FEEM object.
xlab	The x-axis label for the plot, with a sane default.
ylab	The y-axis label for the plot, with a sane default.
cuts	The number of distinct levels the intensity would be divided into, areas between them assined different colours.
col.regions	The palette to take the colours from, a character vector of R colour specifications.
as.table	Whether to draw the panels left to right, top to bottom. (Otherwise they are drawn left to right, bottom to top.)
...	Passed as-is to levelplot.

Value

A **lattice** plot object. Its print or plot method will draw the plot on an appropriate plotting device.

See Also

levelplot

Examples

```
plot(feem(matrix(1:42/42, nrow = 7), 320 + 1:7, 300 + 1:6))
```

[.feem *Extract or replace parts of FEEM objects*

Description

Extract or replace parts of FEEM spectra. Returns FEEM objects unless dimensions should be dropped. When assigning from a FEEM object, requires wavelenths to match and warns if scale factors differ.

Usage

```
  ## S3 method for class 'feem'
x[i, j, drop = TRUE]
  ## S3 replacement method for class 'feem'
x[i, j] <- value
```

Arguments

x	A FEEM object.
i, j	Row and column indices, respectively. As in usual R subsetting, may be integer, logical or character vectors, or missing.
drop	Coerce result to the lowest possible dimension (dropping the feem class if so). This only makes sence for extracting, not replacement.
value	An array-like object to assign values from. When assigning from FEEM objects, wavelengths are required to match and warnings are issued if scale factors don't match.

Value

For [: If isTRUE(drop) and at least one of the index arguments chooses only one element along its axis, a named numeric vector. Otherwise, a FEEM object.

For [<-: a FEEM object.

See Also

feem, [.feemcube

Examples

```
(z <- feem(matrix(1:40, ncol = 8), 66 + 1:5, 99 + 1:8, 3))
str(z[1:4, 1:2])
str(z[1,, drop = TRUE])
z[2:3, 4:5] <- feem(matrix(1:4, 2), 66 + 2:3, 99 + 4:5, 3)
z
```

[.feemcube *Extract or replace parts of FEEM cubes*

Description

Extract or replace single intensities, vectors of them, whole FEEM spectra or even data cubes or their parts from a FEEM cube.

Usage

```
  ## S3 method for class 'feemcube'
x[i, j, k, drop = TRUE]
  ## S3 replacement method for class 'feemcube'
x[i, j, k] <- value
```

Arguments

x	A FEEM cube object.
i, j, k	Row, column and sample indices, respectively. As usual, may be integer, logical or character vectors. Omitting a parameter results in choosing the whole axis.
drop	Coerce result to the lowest possible dimension (dropping the feemcube class). This only makes sence for extracting, not replacement.
value	An array-like object to assign values from. When assigning from FEEM or FEEM cube objects, wavelengths are required to match and warnings are issued if scale factors don't match.

Value

For [: If choosing multiple values along each axis or drop is FALSE, a FEEM cube object. If choosing only one sample but multiple wavelengths, a FEEM object. Otherwise, a named numeric matrix or vector, depending on the dimensions chosen.

For [<-: a FEEM cube object.

See Also

feemcube, [.feem

Examples

```
z <- feemcube(array(1:385, c(5, 7, 11)), 1:5, 1:7, 1:11)
str(z[1:4, 1:2, 1:2])
z[2:3, 4:5, 3] <- feem(matrix(1:4, 2), 2:3, 4:5, 3)
z[,,3]
```

Index