Package: albatross (via r-universe)

September 6, 2024

Type Package

Title PARAFAC Analysis of Fluorescence Excitation-Emission Matrices

Version 0.3-8

Depends R (>= 3.3)

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

Enhances eemR, EEM

Description Perform parallel factor analysis (PARAFAC: Hitchcock, 1927) <doi:10.1002/sapm192761164> on fluorescence excitation-emission matrices: handle scattering signal and inner filter effect, scale the dataset, fit the model; perform split-half validation or jack-knifing. Modified approaches such as Whittaker interpolation, randomised split-half, and fluorescence and scattering model estimation are also available. The package has a low dependency footprint and has been tested on a wide range of R versions.

License GPL (>= 3)

BuildResaveData no

BuildManual yes

NeedsCompilation no

Author Ivan Krylov [aut, cre], Timur Labutin [ths], Anastasia Drozdova [rev]

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

Date/Publication 2024-05-07 15:50:02 UTC

Repository https://ikrylovmsu.r-universe.dev

RemoteUrl https://github.com/cran/albatross

RemoteRef HEAD

RemoteSha 111b0b2e5e0ad869b5d3957dcb3a9f097d06dd69

Contents

`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: handle scattering signal and inner filter effect, scale the dataset, fit the model; perform split-half validation or jack-knifing. Modified approaches such as Whittaker interpolation, randomised split-half, and fluorescence and scattering model estimation are also available. The package has a low dependency footprint and has been tested on a wide range of R versions.

Details

In order to work with your data, create `feem` and/or `feemcube` objects from files or matrix or array objects. Use `feemlist` to import files in bulk. If your files aren't in one of the formats supported by `feem` but you can read their contents by other means, you can supply an importer function to `feemlist`; it should take a file name and return the corresponding `feem` object.

Operations that can be performed on the objects include plotting (`plot.feem`), calculation of fluorescence indices (`feemindex`), inner-filter effect correction (`feemife`), handling of scattering signal (`feemscatter`), changing the wavelength grid of the data by means of interpolation (`feemgrid`), and scaling (`feemscale`). Scaling may be automatically undone after performing the PARAFAC decomposition so that the resulting scores would correspond to the data as it was before the scaling.

All processing functions can take individual `feem` objects, lists of them, or `feemcube` objects and return values of the appropriate kind. For example, `feemscatter` always returns an object of the same class but with the scattering signal handled, while `feemindex` returns named numeric vectors for individual `feem`s but `data.frame`s for collections of them. There's a slight memory benefit to using lists of `feem` objects, but the difference shouldn't be noticeable, so there's nothing to worry about if you started with a `feemcube`.

In order to compute PARAFAC, you need to convert your data into a `feemcube`. Whether you perform jack-knifing, split-half analysis, or PARAFAC itself, a copy of the data cube is kept together with the results and can be extracted back using the `feemcube` function. The resulting objects support a `plot` method (described in the same help page) and can give you the data as a few-column `data.frame` using the `coef` method.

Once the analysis is finished, the PARAFAC model can be exported for the OpenFluor database (`write.openfluor`) or stored as an R object using standard R tools (`save` or `saveRDS`).

Index of help topics:

```
[.feem                  Extract or replace parts of FEEM objects
[.feemcube              Extract or replace parts of FEEM cubes
absindex                Functions of absorbance data
albatross-package       PARAFAC Analysis of Fluorescence
                        Excitation-Emission Matrices
as.data.frame.feem      Transform a FEEM object into a data.frame
feem                    Create a fluorescence excitation-emission
                        matrix object
feemcorcondia           Core consistency diagnostic for PARAFAC models
feemcube                Data cubes of fluorescence excitation-emission
                        matrices
feemflame               Fluorescence and scAttering Model Estimation
feemgrid                Interpolate FEEMs on a given wavelength grid
feemife                 Absorbance-based inner filter effect correction
feemindex               Fluorescence indices and peak values
feemjackknife           Jack-knife outlier detection in PARAFAC models
feemlist                Create lists of FEEM objects
feemparafac             Compute PARAFAC on a FEEM cube object and
                        access the results
feems                   Synthetic 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
marine.colours	Perceptually uniform palettes
plot.feem	Plot a FEEM object
write.openfluor	Export a PARAFAC model for the OpenFluor database

Author(s)

Ivan Krylov [aut, cre], Timur Labutin [ths], Anastasia Drozdova [rev]

References

Murphy KR, Stedmon CA, Graeber D, Bro R (2013). "Fluorescence spectroscopy and multi-way techniques. PARAFAC." *Analytical Methods*, **5**, 6557-6566. doi:10.1039/c3ay41160e.

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

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

Krylov I, Drozdova A, Labutin T (2020). "Albatross R package to study PARAFAC components of DOM fluorescence from mixing zones of arctic shelf seas." *Chemometrics and Intelligent Laboratory Systems*, **207**(104176). doi:10.1016/j.chemolab.2020.104176.

See Also

feem, feemlist, feemindex, feemife, feemscatter, feemgrid, feemcube, feemscale, feemsplithalf, feemparafac, feemjackknife, feemflame, absindex.

Examples

```
data(feems)

dataset <- feemcube(feems, FALSE)
dataset <- feemscatter(dataset, c(24, 15, 10), 'whittaker')
dataset <- feemife(dataset, absorp)
plot(dataset <- feemscale(dataset, na.rm = TRUE))


  # takes a long time
  (sh <- feemsplithalf(
    feemscatter(cube, c(24, 15)),
    # in real life, set a stricter stopping criterion, 1e-6..1e-8
    nfac = 2:4, splits = 4, ctol = 1e-4
  ))
  plot(sh)


pf <- feemparafac(cube, nfac = 3, ctol = 1e-4)
```

```
plot(pf)
```

---

absindex *Functions of absorbance data*

Description

Calculate absorption coefficients and/or absorbance data at given wavelengths, spectral slopes, and their ratios.

Usage

```
absindex(
  x, abs.path, unit = c("log10", "m^-1"), out.A = 254,
  out.a = c(350, 355, 374, 443),
  out.a.ratio = list(c(250, 365), c(465, 665)),
  out.slope = list(c(275, 295), c(350, 400)),
  out.slope.ratio = list(c(275, 295, 350, 400)),
  out.slope.nrmse = FALSE
)
```

Arguments

x	Absorption data, either a list of two-column matrix-like objects with wavelengths in first column and values in second column, or a multi-column matrix-like object with wavelengths in first column and values in all other columns. Can be named in order to match with the abs.path values, see feemife for details.
abs.path	A numeric vector of optical path lengths for every spectrum, in centimetres. Defaults to 1 for all samples. If specified, should be either named, with names matching x, or unnamed and containing exactly the same number of path lengths.
unit	Specifies whether x contains absorbance (log10; $A = \log_{10} \frac{I_0}{I}$; unitless) or absorption coefficients (m^-1; $\alpha = \frac{\ln \frac{I_0}{I}}{l}$; $[\mathrm{m}^{-1}]$).
out.A	Return absorbance values at the wavelengths given as a numeric vector.
out.a	Return absorption coefficients at the wavelengths given as a numeric vector.
out.a.ratio	Return ratios of absorption coefficients at the wavelengths given as a list of two-element numeric vectors. For every pair of wavelengths, $\frac{\alpha(\lambda_1)}{\alpha(\lambda_2)}$ is returned.
out.slope	Return spectral slopes at wavelength ranges given as a list of two-element numeric vectors. See the slope.method for the description of how spectral slopes are computed.
out.slope.ratio	Return ratios of spectral slopes for pairs of wavelength ranges given as a list of four-element numeric vectors. For every list element, the value returned is $\frac{S(\lambda_1,\lambda_2)}{S(\lambda_3,\lambda_4)}$.
out.slope.nrmse	When computing slopes, also return the root-mean-square error for the models providing them, divided by the range of the response: $\frac{1}{y_{\max}-y_{\min}} \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i-y_i)^2}{n}}$.

Details

Currently, the spectral slopes are calculated by fitting a linear model $\ln \alpha = b_0 - b_1 \lambda$ and returning b_1 as the slope. See (Twardowski, Boss, Sullivan, and Donaghay 2004) for a discussion of the calculation methods for spectral slopes.

Requested wavelengths missing from the original grid are interpolated using `spline`. `NA` values are returned outside the original wavelength range.

Value

A `data.frame` with one row per sample, containing the following columns:

`sample` Names or numbers of the samples.

`A.<wavelength>` Absorbance values, for every *wavelength* in `out.A`.

`a.<wavelength>` Absorption coefficients, for every *wavelength* in `out.a`.

`aR.<wl[1]>.<wl[2]>`
Ratios of absorption coefficients, for every *wl* in `out.a.ratio`.

`S.<wl[1]>.<wl[2]>`
Spectral slopes, for every *wl* in `out.slope`.

`NRMSE.S.<wl[1]>.<wl[2]>`
If `out.slope.nrmse` is `TRUE`, root-mean-square errors normalised by the range of the absorption coefficients for the models providing spectral slopes.

`SR.<wl[1]>.<wl[2]>.<wl[3]>.<wl[4]>`
Ratios of spectral slopes, for every *wl* in `out.slope.ratio`.

References

Twardowski MS, Boss E, Sullivan JM, Donaghay PL (2004). "Modeling the spectral shape of absorption by chromophoric dissolved organic matter." *Marine Chemistry*, **89**(1), 69-88. doi:10.1016/j.marchem.2004.02.008.

See Also

`feemife`

Examples

```
data(feems)
absindex(absorp)
```

`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 $(\lambda_{\mathrm{em}}, \lambda_{\mathrm{ex}}, I)$ 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 the column names are already syntactic and the 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 wavelength, nm.
`intensity`	Fluorescence intensity at $(\lambda_{\mathrm{em}}, \lambda_{\mathrm{ex}})$
`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))
```

`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, three-column `data.frame`s containing $(\lambda_{\mathrm{em}}, \lambda_{\mathrm{ex}}, I)$ tuples or files.

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', ...
  )
  ## S3 method for class 'character'
feem(x, format, ...)
  ## S3 method for class 'connection'
feem(x, format, ...)
```

Arguments

`x`	The source of the information to create a FEEM object from: a matrix, a three-column `data.frame`, a file path as a single string, or a connection. If converting a matrix, its rows should correspond to different fluorescence emission wavelengths specified in the `emission` argument; conversely, its columns should correspond to excitation wavelengths specified in the `excitation` argument. If converting a `data.frame`, it should have exactly three columns containing emission wavelengths, excitation wavelength, and intensity values. The names of the columns are expected to be "emission", "excitation", and "intensity", respectively, but can be overridden using namesake arguments. If reading a single file by file path or connection, the `format` argument must specify the kind of file to parse, see below.
`emission`	If converting 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 converting 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 divide the resulting score values after running PARAFAC. If the EEM has been pre-multiplied prior to creating the FEEM object, you can set the multiplier here.

`format` **table** The FEEM is assumed to be stored as a plain text matrix, readable using `read.table`, with the first column and the first row containing wavelengths. For example, it is possible to import CSV files obtained from a HORIBA Aqualog® fluorometer by using `feem(`*file*`, 'table', sep = ',')`.
Rows are assumed to correspond to emission wavelengths, columns are assumed to correspond to excitation wavelengths; if that's not the case, set the `transpose` argument to `TRUE`.
If there are unmeasured points in the spectrum (e.g. the anti-Stokes area) encoded as special numbers (e.g. stored as zeroes or -100), specify their values as the `na` argument (numeric vector). The function will check for triangles filled with these values (such that a threshold Δ exists where for all $\lambda_{\mathrm{em}} - \lambda_{\mathrm{ex}} > \Delta$ or $\lambda_{\mathrm{em}} - \lambda_{\mathrm{ex}} < \Delta$, $X(\lambda_{\mathrm{em}}, \lambda_{\mathrm{ex}}) \in$ `na`) and replace them with `NA`s. If the unmeasured values are not stored as numbers, use the `na.strings` argument of `read.table` to specify them.
The `fileEncoding` argument is treated specially (but preserving the `read.table` semantics regarding numbers), preventing errors in case the session charset cannot represent all characters from the file: the contents are first converted to UTF-8, then forwarded to `read.table`. This only works when `file` is a file path, not a connection. (When passing a connection with an encoding attribute set, make sure that the file contents after decoding can be represented in the session charset.)
The `dec` argument is also treated specially, making it possible to read a column containing both numbers (with a decimal comma) and strings and then convert them to numbers (transforming the strings into `NA_real_`).
All other arguments are passed to `read.table`, with `fill` defaulting to `TRUE` instead of `FALSE`.

panorama Read a '`.dat`' file as created by "Panorama" software that comes with FLUORAT®-02-PANORAMA fluorometer. Such files contain a header describing the wavelength range, e.g.:

```
240.0   650.0     1.0    ; Emission(columns)
230.0   320.0     5.0    ; Excitation(rows)
```

The header is followed by the intensity data as matrix, whitespace-separated. Missing points (anti-Stokes area) stored as 0 or 100 and are automatically filtered out on import. No additional parameters are accepted.

F900txt Read an emission map text file created by "F900" software that comes with Edinburgh Instruments fluorescence spectrometers. Separated by empty lines, these files are assumed to contain:

1. Sample name

2. Metadata of each emission scan, with names in the first column, including the excitation wavelengths
3. Fluorescence values with emission wavelengths in the first column

The `fileEncoding` argument specifies the encoding of the text file and has a safe default of "latin1". It should typically correspond to the ANSI code page on the computer running F900 that was used to create the file. See `iconvlist()` for a list of encodings understood by R.

The `sep` argument specifies the column separator used in the file. A semicolon `;` is typically used, though other options are possible.

`...` When converting matrices and `data.frame`s, extra arguments besides those specified above are not allowed.

When reading the FEEM from a file, additional arguments may be passed to `format`-specific reading functions, see above.

Details

Transposing a `feem` object using `t` will remove the class attribute, returning an ordinary matrix.

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`, `feemgrid`, `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'
)
feem(
  system.file('extdata/ho_aq.csv', package = 'albatross'),
  'table', sep = ','
)
feem(
  system.file('extdata/F900.txt', package = 'albatross'), 'F900txt'
)
```

`feemcorcondia` *Core consistency diagnostic for PARAFAC models*

Description

Compute the core consistency diagnostic ("CORCONDIA") by fitting a "Tucker3" core array to the existing PARAFAC loadings.

Usage

```
  feemcorcondia(
    model, divisor = c("nfac", "core"),
    kind = c('pinv', 'iterative', 'vec'), ...
  )
  ## S3 method for class 'feemcorcondia'
print(x, ...)
```

Arguments

`model`	A PARAFAC model returned by `feemparafac`.
`divisor`	The divisor used in computation of the CORCONDIA value, see Details.
`kind`	A string choosing the method used to compute the least squares Tucker3 core. Defaults to "pinv" for PARAFAC models without missing data and "iterative" for models where missing data is present. See Details.
`x`	An object returned by `feemcorcondia`.
`...`	`feemcorcondia` For `kind = 'iterative'`, forwarded to `optim` (see Details). Otherwise, not allowed. `print.feemcorcondia` Ignored.

Details

The "Tucker3" model uses three loading matrices and a small three-way "core array" to describe a larger three-way array:

$$X_{i,j,k} = \sum_r \sum_s \sum_t A_{i,r} B_{j,s} C_{k,t} G_{r,s,t}$$

It's easy to show that constraining $G_{r,s,t} = 1_{r=s=t}$ makes the Tucker3 model equivalent to a PARAFAC model. The core consistency diagnostic works by constraining the loading matrices of a Tucker3 model to the existing loading matrices from a PARAFAC model and estimating the core array. The closer the resulting $\mathbf{G}$ tensor is to a diagonal one, the better.

Given the least-squares estimated core tensor $\mathbf{G}$, the ideal core tensor $T_{r,s,t} = 1_{r=s=t}$ and the denominator D, the CORCONDIA metric is defined as follows:

$$\left(1 - \frac{\sum_r \sum_s \sum_t (G_{r,s,t} - T_{r,s,t})^2}{D}\right) \cdot 100\%$$

The denominator can be chosen to be either $\sum_r \sum_s \sum_t T^2_{r,s,t}$, which is equal to the number of factors in the model (`divisor = 'nfac'`), or $\sum_r \sum_s \sum_t G^2_{r,s,t}$, which will avoid resulting negative values (`divisor = 'core'`).

There are multiple ways how the least squares Tucker3 core can be computed. When no data is missing, the matricised form of the model can be used to derive the expression (`kind = 'pinv'`, the default in such cases):

$$\mathbf{X} = \mathbf{A}\mathbf{G}(\mathbf{C} \otimes \mathbf{B})^\top + \epsilon$$
$$\hat{\mathbf{G}} = \mathbf{A}^+\mathbf{X}((\mathbf{C}^\top)^+ \otimes (\mathbf{B}^\top)^+)$$

With missing data present, a binary matrix of weights $\mathbf{W}$ appears:

$$\min_{\mathbf{G}} \left\| \mathbf{W} \circ (\mathbf{A}\mathbf{G}(\mathbf{C} \otimes \mathbf{B})^\top - \mathbf{X}) \right\|^2$$

A gradient-based method can be used to solve this problem iteratively without allocating too much memory, but care must be taken to ensure convergence. For `kind = 'iterative'` (which is the default for models with missing data), `optim` is used with parameters `method = 'L-BFGS-B'`, `control = list(maxit = 1000, factr = 1)`. Warnings will be produced if the method doesn't indicate successful convergence.

The problem can also be solved exactly by unfolding the tensor into a vector and skipping the elements marked as missing:

$$\min_{\mathbf{G}} \left\| \text{vec}(\mathbf{A}\mathbf{G}(\mathbf{C} \otimes \mathbf{B})^\top)_{\text{non-missing}} - \text{vec}(\mathbf{X})_{\text{non-missing}} \right\|^2$$
$$\text{vec}(\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})_{\text{non-missing}} \times \text{vec}\, \mathbf{G} = \text{vec}(\mathbf{X})_{\text{non-missing}}$$

Unfortunately, when this method is used (`kind = 'vec'`), the left-hand side of the equation has the size of $\mathbf{X}$ times the number of components cubed, which grows very quickly for models with large numbers of components.

Value

A numeric scalar of class `feemcorcondia` with the following attributes:

`divisor`	The `divisor` argument, expanded to one of the valid options.
`core`	A three-way array containing the least-squares Tucker core for the given PARAFAC model.

References

Bro R, Kiers HAL (2003). "A new efficient method for determining the number of components in PARAFAC models." *Journal of Chemometrics*, **17**(5), 274-286. ISSN 0886-9383, doi:10.1002/cem.801.

See Also

`multiway::corcondia`

Examples

```
data(feems)
cube <- feemscale(feemscatter(cube, c(20, 14)), na.rm = TRUE)
# kind = 'vec' is exact but may take a lot of memory
feemcorcondia(feemparafac(cube, nfac = 3, ctol = 1e-4), kind = 'vec')
# kind = 'iterative' used by default for models with missing data
feemcorcondia(feemparafac(cube, nfac = 4, ctol = 1e-4))
```

---

feemcube *Data cubes of fluorescence excitation-emission matrices*

Description

Given a list of feem objects or a 3-way array, build tagged 3-dimensional arrays of fluorescence excitation-emission spectra. Extract the data cube from the corresponding model objects. Transform the data cube into a list of feem objects.

Usage

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

Arguments

x	**feemcube** A list of FEEM objects, possibly named, or a numeric array. Alternatively, a feemparafac, feemjackknife, feemsplithalf, or a feemflame object. **as.list.feemcube** A feemcube object.
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 may be set to NA (all.wavelengths = TRUE) or omitted from the cube (all.wavelengths = FALSE). See feemgrid if you need to adjust the wavelength grid of a list of EEMs before making it into a FEEM cube.

feemcube.feemparafac, feemcube.jackknife, and feemcube.feemsplithalf return the data cube originally passed to the corresponding functions.

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 if not specified by the user.

as.list.feemcube: A named list of FEEM objects comprising x.

See Also

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

Examples

```
# array form
feemcube(
  array(1:24, c(4, 3, 2)), # 3-way array obtained elsewhere
  seq(340, 400, len = 4), seq(250, 300, len = 3) # wavelengths
)
# list form
feemcube(
  replicate(2, feem( # list of feem objects
    matrix(1:6, 2), c(340, 400), c(250, 275, 300)
  ), FALSE),
```

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

---

feemflame *Fluorescence and scAttering Model Estimation*

Description

Given a FEEM cube, model the fluorescence and the scattering signals at the same time as a sum of a PARAFAC model and a low-rank unfolded matrix factorisation.

Usage

```
  feemflame(
    X, ffac, sfac, maxiter = 32, widths = rep(25, 4), Raman.shift = 3400,
    ctol = 1e-04, progress = TRUE, control.parafac, control.cmf
  )
  ## S3 method for class 'feemflame'
fitted(object, ...)
  ## S3 method for class 'feemflame'
residuals(object, ...)
  ## S3 method for class 'feemflame'
coef(
    object, type = c(
      "fluorescence",
      "scores", "loadings", "emission", "excitation", "samples",
      "scattering", "sc.scores", "sc.loadings"
    ), ...
  )
  ## S3 method for class 'feemflame'
plot(
    x, type = c('both', 'fl.image', 'fl.lines'), ...
  )
```

Arguments

X	A feemcube object.
ffac	The number of trilinear components used to model fluorescence, passed to feemparafac.
sfac	The number of bilinear (low-rank matrix factorisation) components used to model the scattering signal.
maxiter	Maximum number of alternating PARAFAC and constrained matrix factorisation iterations.
widths	Widths of the scattering regions, like in feemscatter: 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

`Raman.shift` Raman shift of the scattering signal, in cm^{-1}, like in `feemscatter`.

`ctol` Given $L = ||\mathbf{X} - \hat{\mathbf{X}}||^2$, stop when $\frac{|\Delta L|}{L} \leq$ `ctol`.

`progress` Print progress information on the console, including the iteration number, relative sum of squared residuals, and relative change in sum of squared residuals.

`control.parafac`, `control.cmf`
Named lists of additional arguments to be passed to the underlying functions. Both default to `list(ctol = 1e-4, maxit = 10)`, which makes both steps to run for 10 iterations or stop when the absolute change in R^2 is less than 10^{-4}, whichever happens sooner.

`object`, `x` A `feemflame` object.

`type` **coef** Determines the type of coefficients to return:

fluorescence Equivalent to calling `coef.feemparafac` on the fluorescence model (default).

scores, loadings, emission, excitation, samples Equivalent to calling `coef.feemparafac` on the fluorescence model and passing the respective `type` argument.

sc.scores A `data.frame` containing the following columns:

sample Sample numbers or names.

value Scattering intensity value for a given factor.

factor The number of the scattering component.

sc.loadings A `data.frame` containing the following columns:

emission, excitation The wavelengths corresponding to the value of the scattering profile.

value Scattering intensity value for a given factor.

factor The number of the scattering component.

scattering A list with names "scores" and "loadings" containing results of `coef(object, 'sc.scores')` and `coef(object, 'sc.loadings')`, respectively.

plot Describes the kind of plot to produce:

both Plot the loadings of the fluorescence and scattering models as false colour images.

fl.image, fl.lines Equivalent to calling `plot.feemparafac` on the fluorescence model with the argument "image" or "lines", respectively.

`...` No other parameters are allowed.

Details

FLAME models the input data as a sum of fluorescence signal (PARAFAC model) and scattering signal (low rank model):

$$X_k(\lambda_i^{\mathrm{em}}, \lambda_j^{\mathrm{ex}}) = \underbrace{\sum_p A_{i,p} B_{j,p} C_{k,p}}_{\text{fluorescence}} + \underbrace{\sum_q S_{i,j,q} D_{k,q}}_{\text{scattering}}$$

The function alternates between fitting the PARAFAC model on the dataset with scattering signal subtracted and fitting the low-rank model on the dataset with fluorescence signal subtracted. The PARAFAC model is fitted using the `feemparafac` function. The low-rank model is fitted by means of unfolding the wavelength dimensions into one, resulting in a matrix, followed by the same alternating least squares procedure as done in multivariate curve resolution. Both models are constrained to result in non-negative factors.

The low-rank model is additionally constrained to zero outside the scattering region. The scattering region is defined the same way as in `feemscatter`, using the `widths` and the `Raman.shift` arguments.

Initial PARAFAC model is fitted with the scattering region set to missing. The low-rank model is initialised with truncated singular value decomposition forced to be non-negative.

Value

`feemflame` An object of class `feemflame`, which is a list containing the following components:

fl A `feemparafac` object containing the fluorescence part of the model.

sc An object of internal class `cmf`. Please don't rely on its structure.

`fitted.feemflame`
A `feemcube` object containing the part of `X` fitted by the model.

`residuals.feemparafac`
A `feemcube` object equal to $\mathbf{X} - \hat{\mathbf{X}}$, with an extra class `feem.resid` set. Objects of this class are plotted with a different default palette, see `plot.feem.resid`.

`coef.feemflame` See the description of the `type` argument.

Note

The structure of the `feemflame` object, the initialisation, and the constraints may be subject to change in a future version.

References

Tauler R, Marqués I, Casassas E (1998). "Multivariate curve resolution applied to three-way trilinear data: Study of a spectrofluorimetric acid-base titration of salicylic acid at three excitation wavelengths." *Journal of Chemometrics*, **12**(1), 55-75. doi:10.1002/(SICI)1099128X(199801/02)12:1<55::AIDCEM501>3.0.CO;2#.

Krylov I, Labutin T, Rinnan Å, Bro R (2021). "Modelling of scattering signal for direct PARAFAC decompositions of excitation-emission matrices." 17th Scandinavian Symposium on Chemometrics. https://web.archive.org/web/20220314144225/https://ssc17.org/abstract/Krylov1.html. https://files.libs.chem.msu.ru/~ivan/SSC17/P13.pdf.

See Also

feemparafac, feemcube

Examples

```
data(feems)
cube <- feemscale(cube)
factors <- feemflame(cube, ffac = 3, sfac = 1)
str(coef(factors))
str(coef(factors, 'scattering'))
plot(factors)
```

---

feemgrid *Interpolate FEEMs on a given wavelength grid*

Description

Use interpolation to change the wavelength grid of a single FEEM or unify the grid of a collection of them.

Usage

```
  feemgrid(x, ...)
  ## S3 method for class 'feem'
feemgrid(
    x, emission, excitation,
    method = c("whittaker", "loess", "kriging", "pchip"), ...
  )
  ## S3 method for class 'feemcube'
feemgrid(
    x, emission, excitation, ..., progress = TRUE
  )
  ## S3 method for class 'list'
feemgrid(
    x, emission, excitation, ..., progress = TRUE
  )
```

Arguments

x	A feem object, a feemcube, or a list of feem objects.
emission, excitation	Desired wavelength grid, as numeric vectors. Must be specified for a single FEEM. If not specified for a collection of FEEMs, all wavelengths falling in the range of the intersection all wavelengths intervals are chosen.
method	Interpolation method, see feemscatter for details.
...	Passed from generics to feemgrid.feem, then to interpolation methods. See feemscatter for details.
progress	Set to FALSE to disable the progress bar.

Details

The algorithm doesn't know how to distinguish between NAs that haven't been measured and NAs that resulted from combining different wavelength grids, so it tries to interpolate all of them. As a result, leaving large areas of the spectrum undefined (e.g. anti-Stokes area) is not recommended, since it would result in extrapolation and introduce strong artefacts.

Value

An object of the same kind (FEEM object / FEEM cube / list of them) with emission and excitation wavelengths as requested.

See Also

feemscatter

Examples

```
data(feems)
x <- feemscatter(feems$a, rep(25, 4))
y <- feemgrid(x, seq(240, 600, 5), seq(230, 550, 10))
plot(plot(x, main = 'Original'    ), split = c(1, 1, 2, 1), more = TRUE)
plot(plot(y, main = 'Interpolated'), split = c(2, 1, 2, 1))
```

---

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, ..., progress = FALSE)
  ## S3 method for class 'list'
feemife(x, absorbance, abs.path, ..., progress = FALSE)
```

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 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 object. 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 x or absorbance isn't named, but (given N-sample x) absorbance has exactly $N + 1$ columns or is an N-element list, absorbance spectra are assumed to be present in the same order as the samples in x. Otherwise, an error is raised.

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.

progress — Set to TRUE to enable a progress bar (implemented via txtProgressBar).

... — No parameters besides those described above are allowed.

Details

If you receive errors alleging that some names don't match, but are absolutely sure that the absorbance spectra and path lengths are present in the same order as in x, remove the names from either of the objects, e.g. by passing unname(absorbance).

The formula used to correct for inner filter effect is:

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

Value

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

References

Lakowicz JR (2006). *Principles of Fluorescence Spectroscopy, 3rd ed.*. Springer US. doi:10.1007/9780387463124.

Kothawala DN, Murphy KR, Stedmon CA, Weyhenmeyer GA, Tranvik LJ (2013). "Inner filter correction of dissolved organic matter fluorescence." *Limnology and Oceanography: Methods*, **11**(12), 616-630. doi:10.4319/lom.2013.11.616.

Examples

```
data(feems)

str(cube)
str(absorp)
plot(feemife(cube,absorp) / cube)
```

`feemindex` *Fluorescence indices and peak values*

Description

Calculate fluorescence indices or peak values for individual FEEMs or groups of them.

Usage

```
  feemindex(x, ...)
  ## S3 method for class 'feem'
feemindex(
    x,
    indices = c(
      "HIX", "BIX", "MFI", "CFI", "YFI", "FrI",
      "A", "B", "C", "M", "P", "T"
    ),
    tolerance = 1, interpolate = FALSE, ...
  )
  ## S3 method for class 'feemcube'
feemindex(x, ..., progress = FALSE)
  ## S3 method for class 'list'
feemindex(x, ..., progress = FALSE)
```

Arguments

`x`	A FEEM, a FEEM cube, or a list of `feem` objects.
`indices`	Fluorescence indices or peaks to return. By default, all indices and peaks known to the function are returned. See Details for their meaning.
`tolerance`	A numeric scalar signifying the acceptable emission and excitation wavelength error in nm. For example, if a wavelength of 254 nm is needed to calculate an index, a value at 255 nm can be considered if `tolerance >= 1`. Defaults to 1 nm. See below for what happens if no matching value is found.
`interpolate`	A string specifying an interpolation method ("whittaker", "loess", "kriging", "pchip"), or `FALSE` to disable interpolation (default). If interpolation is disabled, an index will get an `NA` value when required points are too far from the measured grid or are present in the grid but set to `NA`. When interpolation is enabled, required points that are missing from the grid or present but set to `NA` will be interpolated using `feemgrid` as long as they are within the wavelength bounds of the FEEM. `NA`s may still be returned only when the desired value is impossible to interpolate due to it being outside the wavelength range.
`...`	Additional parameters eventually passed to interpolation methods. See `feemscatter` for details.
`progress`	Set to `TRUE` to enable a progress bar (implemented via `txtProgressBar`).

Details

Available indices and peaks are:

HIX

$$\mathrm{HIX} = \frac{\int_{435\,\mathrm{nm}}^{480\,\mathrm{nm}} I\, d\lambda_\mathrm{em}}{\int_{300\,\mathrm{nm}}^{345\,\mathrm{nm}} I\, d\lambda_\mathrm{em}} \text{ at } \lambda_\mathrm{ex} = 254\,\mathrm{nm}$$

Higher values of the humification index correspond to more condensed fluorescing molecules (higher C/H), more humified matter. (Zsolnay, Baigar, Jimenez, Steinweg, and Saccomandi 1999)

BIX

$$\mathrm{BIX} = \frac{I(\lambda_\mathrm{em} = 380\,\mathrm{nm})}{I(\lambda_\mathrm{em} = 430\,\mathrm{nm})} \text{ at } \lambda_\mathrm{ex} = 310\,\mathrm{nm}$$

Index of recent autochthonous contribution determines the presence of the β fluorophore, characteristic of autochthonous biological activity in water samples. (Huguet, Vacher, Relexans, Saubusse, Froidefond, and Parlanti 2009)

MFI

$$\mathrm{MFI} = \frac{I(\lambda_\mathrm{em} = 450\,\mathrm{nm})}{I(\lambda_\mathrm{em} = 500\,\mathrm{nm})} \text{ at } \lambda_\mathrm{ex} = 370\,\mathrm{nm}$$

The fluorescence index by (McKnight, Boyer, Westerhoff, Doran, Kulbe, and Andersen 2001) helps distinguish sources of isolated aquatic fulvic acids and may indicate their aromaticity.

CFI

$$\mathrm{CFI} = \frac{I(\lambda_\mathrm{em} = 470\,\mathrm{nm})}{I(\lambda_\mathrm{em} = 520\,\mathrm{nm})} \text{ at } \lambda_\mathrm{ex} = 370\,\mathrm{nm}$$

The fluorescence index by (Cory and McKnight 2005) is correlated to relative contribution of microbial versus higher plant-derived organic matter to the DOM pool.

YFI

$$\mathrm{YFI} = \frac{\bar{I}(\lambda_\mathrm{em} \in [350, 400]\,\mathrm{nm})}{\bar{I}(\lambda_\mathrm{em} \in [400, 450]\,\mathrm{nm})} \text{ at } \lambda_\mathrm{ex} = 280\,\mathrm{nm}$$

Yeomin fluorescence index (Heo, Yoon, Kim, Lee, Lee, and Her 2016) is lowest for humic-like and fulvic-like samples, higher for aminosugar-like samples and highest for protein-like samples.

FrI

$$\mathrm{FrI} = \frac{I(\lambda_\mathrm{em} = 380\,\mathrm{nm})}{\max I(\lambda_\mathrm{em} \in [420, 435]\,\mathrm{nm})} \text{ at } \lambda_\mathrm{ex} = 310\,\mathrm{nm}$$

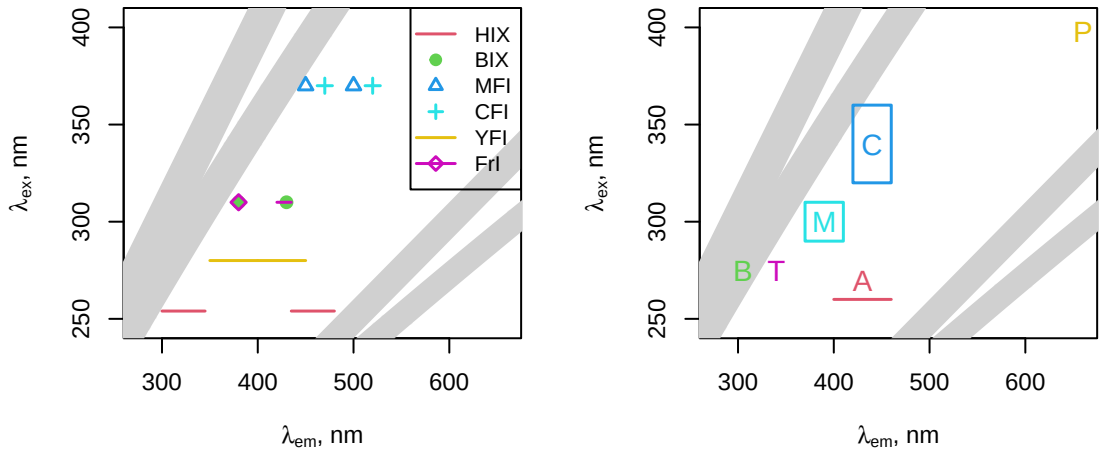
The freshness index, also known as $\frac{\beta}{\alpha}$, is an indicator of autochthonous inputs (Wilson and Xenopoulos 2009) and may provide indication of relative contribution of microbially produced DOM.

A, B, C, M, P, T Fluorophore peaks taken from (Coble 2007):

Peak	λ_ex	λ_em	Fluorescence
A	260	400-460	humic-like
B	275	305	tyrosine-like
C	320-360	420-460	humic-like

M	290-310	370-410	marine humic-like
P	398	660	pigment-like
T	275	340	tryptophan-like

When a range of wavelengths specified in one or both axes, the maximal signal value over that range is taken.



Positions of the peaks and the areas used to determine the fluorescence indices of an EEM. The Rayleigh and Raman scattering areas for both 1st and 2nd diffraction orders are shown in grey, assuming a width of ± 20 nm and a Raman shift of $3400\ \mathrm{cm}^{-1}$. The tolerance interval of ± 1 nm is invisible at the scale of the figure.

Integration for HIX and YFI is done using the trapezoidal method:

$$\int_a^b f(x)dx \approx (b-a)\frac{f(a)+f(b)}{2}$$

Value

For individual `feem` objects, a named numeric vector containing the values requested via the `indices` argument.

Otherwise, a `data.frame` containing the values from the vectors above and a column named `sample` containing the names of the samples (or numbers, if names were absent).

Author(s)

With edits and suggestions by Anastasia Drozdova.

References

Coble PG (2007). "Marine Optical Biogeochemistry: The Chemistry of Ocean Color." *Chemical Reviews*, **107**(2), 402-418. doi:10.1021/cr050350+.

Cory RM, McKnight DM (2005). "Fluorescence spectroscopy reveals ubiquitous presence of oxidized and reduced quinones in dissolved organic matter." *Environmental science & technology*, **39**(21), 8142-8149. doi:10.1021/es0506962.

Heo J, Yoon Y, Kim D, Lee H, Lee D, Her N (2016). "A new fluorescence index with a fluorescence excitation-emission matrix for dissolved organic matter (DOM) characterization." *Desalination and Water Treatment*, **57**(43), 20270-20282. doi:10.1080/19443994.2015.1110719.

Huguet A, Vacher L, Relexans S, Saubusse S, Froidefond JM, Parlanti E (2009). "Properties of fluorescent dissolved organic matter in the Gironde Estuary." *Organic Geochemistry*, **40**(6), 706-719. doi:10.1016/j.orggeochem.2009.03.002.

McKnight DM, Boyer EW, Westerhoff PK, Doran PT, Kulbe T, Andersen DT (2001). "Spectrofluorometric characterization of dissolved organic matter for indication of precursor organic material and aromaticity." *Limnology and Oceanography*, **46**(1), 38-48. doi:10.4319/lo.2001.46.1.0038.

Wilson HF, Xenopoulos MA (2009). "Effects of agricultural land use on the composition of fluvial dissolved organic matter." *Nature Geoscience*, **2**(1), 37-41. doi:10.1038/ngeo391.

Zsolnay A, Baigar E, Jimenez M, Steinweg B, Saccomandi F (1999). "Differentiating with fluorescence spectroscopy the sources of dissolved organic matter in soils subjected to drying." *Chemosphere*, **38**(1), 45-50. doi:10.1016/S00456535(98)001660.

See Also

`feem`

Examples

```
data(feems)

x <- feemscatter(feems$a, rep(25, 4), 'omit')
feemindex(x)
feemindex(x, interpolate = 'whittaker')

feemindex(feems[2:3])
feemindex(feemcube(feems[4:5], TRUE))
```

---

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

```
  )
  ## S3 method for class 'feemjackknife'
coef(
    object, kind = c('estimations', 'RIP', 'IMP'), ...
  )
```

Arguments

cube	A feemcube object.
progress	Set to FALSE to disable the progress bar.
x, object	An object returned by feemjackknife.
kind	Chooses what to plot (when called as plot(...)) or return as a data.frame (when called as coef(...)):
	estimations Produce the loadings from every leave-one-out model.
	RIP Produce a Resample Influence Plot, i.e. mean squared difference between loadings in overall and leave-one-out models plotted against mean squared residuals in leave-one-out models.
	IMP Produce an Identity Match Plot, i.e. scores in leave-one-out models plotted against scores in the overall model.
...	**feemjackknife** Passed as-is to feemparafac and, eventually, to **multiway** function 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. Default value for q is 0.9. Remaining arguments are passed as-is to xyplot.
	coef.feemjackknife No further parameters are allowed.

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:

$$\hat{\mathbf{c}} = (\mathbf{A} * \mathbf{B})^{+} \times \mathrm{vec}(\mathbf{X})$$

The individual leave-one-out models (fitted loadings $\mathbf{A}$, $\mathbf{B}$ and scores $\mathbf{C}$) 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 $\mathrm{diag}(\mathbf{s}_\mathrm{A}) \times \mathrm{diag}(\mathbf{s}_\mathrm{B}) \times \mathrm{diag}(\mathbf{s}_\mathrm{C}) = \mathbf{I}$, 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.

To conserve memory, feemjackknife puts the user-provided cube in an environment and passes it via envir and subset options of feemparafac. This means that, unlike in feemparafac, the cube argument has to be a feemcube object and passing envir and subset options to feemjackknife is not supported. 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 A 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.

coef.feemjackknife A `data.frame` containing various columns, depending on the value of the `kind` argument:

- **estimations loading** Values of the loadings.
  - **mode** The axis of the loadings, "Emission" or "Excitation".
  - **wavelength** Emission or excitation wavelength the loading values correspond to.
  - **factor** The component number.
  - **omitted** The sample (name if `cube` had names, integer if it didn't) that was omitted to get the resulting loading values.
- **RIP msq.resid** Mean squared residual value for the model with a given sample omitted.
  - **Emission** Mean squared difference in emission mode loadings between the overall model and the model with a given sample omitted.
  - **Excitation** Mean squared difference in excitation mode loadings between the overall model and the model with a given sample omitted.
  - **omitted** The sample (name if `cube` had names, integer if it didn't) that was omitted from a given model.
- **IMP score.overall** Score values for the overall model.
  - **score.predicted** Score values estimated from the loadings of the model missing a given sample.
  - **factor** The component number.
  - **omitted** The sample (name if `cube` had names, integer if it didn't) that was omitted from a given model.

References

Riu J, Bro R (2003). "Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models." *Chemometrics and Intelligent Laboratory Systems*, **65**(1), 35-49. doi:10.1016/S01697439(02)000904.

See Also

`feemparafac`, `feemcube`

Examples

```
data(feems)
cube <- feemscale(feemscatter(cube, rep(14, 4)), na.rm = TRUE)
# takes a long time; the stopping criterion is weaked for speed
jk <- feemjackknife(cube, nfac = 3, ctol = 1e-4)
# feemparafac methods should be able to use the environment and subset
plot(jk$leaveone[[1]])
plot(jk)
plot(jk, 'IMP')
plot(jk, 'RIP')
head(coef(jk))
```

`feemlist` *Create lists of FEEM objects*

Description

Convert vectors of file names or objects from other packages (such as **eemR** or **EEM**) into flat named lists of `feem` objects.

Usage

```
  feemlist(x, ...)
  ## S3 method for class 'character'
feemlist(
    x, format, pattern = NULL, recursive = TRUE, ignore.case = FALSE,
    simplify.names = TRUE, progress = TRUE, ...
  )
  ## S3 method for class 'eemlist'
feemlist(x, ...)
  ## S3 method for class 'EEM'
feemlist(x, ...)
```

Arguments

`x`	A character vector containing names of files and directories to import using `feem`. Alternatively, an `eemlist` object from the **eemR** package or an `EEM` object from the **EEM** package.
`format`	Corresponds to the `format` argument of `feem`. Currently, one format is assumed for all files to be imported. Alternatively, can be a function that takes a path to a file and anything passed in `...` and returns a `feem` object corresponding to the file. This is done to make it easier to import groups of files in formats not yet supported by `feem` itself.
`pattern, recursive, ignore.case`	These options are passed to `list.files` for directories encountered in `x` and can be used to e.g. only choose files with a given suffix in the name. Note the non-default value for the `recursive` option.

simplify.names	If TRUE (default), split resulting names by the path separator (/, also \ on Windows) and remove leading components that have the same value for all samples, but leave at least one component. See Details on how this is related to name generation.
progress	If true, show a text progress bar using txtProgressBar.
...	When importing files, remaining options are passed to feem or the import function passed in the format argument. Otherwise, no options are allowed.

Details

Names of x are preserved; if x is not named, names are assigned from the values of x itself, and so are empty names in partially-named x. Every directory in x is replaced with its contents (as returned by list.files), their names obtained by concatenating the name of the directory element with their paths inside the directory (with .Platform$file.sep as a separator). For example, when importing x = c('foo' = 'bar') with directory 'bar' containing 'baz.txt', resulting name would be 'foo/baz.txt'.

When importing many files from the same directory, the simplify.names option is useful to avoid duplication in resulting names. For example, feemlist('.', simplify.names = FALSE) results in a list with all names starting with ./, while feemlist('foo/bar/baz', simplify.names = TRUE) (default) would shave off all three common path components and the separators.

Mixing files and directories in x will most likely not preserve the order of the elements.

Note: Please don't rely on the name generation behaving exactly as specified as it may be changed in the future versions.

When importing custom file formats, the format function should typically take the following form:

```
function(filename, ...) {
  # read data from filename
  # take additional arguments passed from feemlist(...) if needed
  return(feem(data))
}
```

Value

A flat named list of feem objects.

See Also

feem; the packages **eemR** and **EEM**.

Examples

```
feemlist(
  system.file('extdata/pano2.txt', package = 'albatross'),
  'table', transpose = TRUE, na = 0
)
if (requireNamespace('eemR')) feemlist(eemR::eem_read(
  system.file('extdata/ho_aq.csv', package = 'albatross'),
  import_function='aqualog'
```

```
))
if (requireNamespace('EEM')) feemlist(EEM::readEEM(
  system.file('extdata/ho_aq.dat', package = 'albatross')
))
feemlist(
  system.file('extdata/custom.rds', package = 'albatross'),
  readRDS
)
```

`feemparafac` *Compute PARAFAC on a FEEM cube object and access the results*

Description

`feemparafac` forwards its arguments to `parafac` from the **multiway** package, optionally rescales the result and attaches a few attributes. Resulting objects of class `feemparafac` can be accessed using methods presented below.

Usage

```
  feemparafac(
    X, ..., const = rep('nonneg', 3), ctol = 1e-6,
    rescale = 3, retries = 10, subset = TRUE, envir = NULL
  )
  ## S3 method for class 'feemparafac'
plot(x, type = c("image", "lines"), ...)
  ## S3 method for class 'feemparafac'
coef(
    object, type = c(
      "all", "scores", "loadings", "surfaces",
      "emission", "excitation", "samples"
    ), ...
  )
  ## S3 method for class 'feemparafac'
fitted(object, ...)
  ## S3 method for class 'feemparafac'
residuals(object, ...)
  ## S3 method for class 'feemparafac'
reorder(x, neworder, like, ...)
  ## S3 method for class 'feemparafac'
rescale(x, mode, newscale, absorb, like, ...)
```

Arguments

`X`	A FEEM cube object. The per-sample factors will be multiplied by `attr(X, 'scales')` stored in it. If `envir` is `NULL` (by default), this should be just a value. If `envir` is given, this should be a name of the value to `get` from the environment.

`...`

feemparafac Passed as-is to `parafac`.

plot.feemparafac Passed as-is to **lattice** functions `levelplot` and `xyplot`.

reorder.feemparafac, rescale.feemparafac Forwarded to the respective **multiway** functions.

coef.feemparafac, fitted.feemparafac, residuals.feemparafac No other parameters are allowed.

`const` A character vector of length 3 specifying the constraints for all modes of `X`, passed to `parafac`. Defaults to non-negativity. See `const` for more information.

`ctol` The stopping criterion used by `parafac`. When a step results in an absolute change in R^2 below `ctol`, the algorithm stops.

`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.

`subset` An integer or logical vector choosing the samples from `X`, as in `feemparafac(X[,,subset], ...)`. Defaults to the whole cube.

`envir` An environment to look up `X` in.

`x`, `object` An object returned by `feemparafac`.

`type` Given a fitted PARAFAC model:

$$X_{i,j,k} = \sum_r A_{i,r} B_{j,r} C_{k,r}$$

With $\mathbf{A}$ corresponding to fluorescence emission loadings, $\mathbf{B}$ corresponding to fluorescence excitation loadings, 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 pseudo-colour images of outer products $\mathbf{a}_r \times \mathbf{b}_r^\top$

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

Fitted PARAFAC coefficients can be returned in the following forms:

emission, excitation, samples Return the contents of $\mathbf{A}$, $\mathbf{B}$ or $\mathbf{C}$, respectively, as a `data.frame` with three columns, the first one (named `wavelength` or `sample`) containing the independent variable (λ_{em} / λ_{ex} / sample name or number), the second one (named `value`) containing the values and the third one (named `factor`) containing the factor numbers.

scores Same as samples.

loadings Same as "emission" and "excitation" combined using `rbind`, with a fourth column (`mode`) added, naming the kinds of loadings.

all A list with names "emission", "excitation", "samples" containing results of `coef(object, "emission")`, `coef(object, "excitation")`, `coef(object, "samples")`, respectively.

surfaces A `data.frame` containing the columns `emission`, `excitation` (containing the wavelengths), `intensity` (containing the values of the outer product $\mathbf{a}_r \times \mathbf{b}_r^\top$), and `factor` (containing the factor numbers r).

`neworder` A permutation of integers between 1 and `ncol(x$A)` (the number of components) specifying the new order of factors. Forwarded to `reorder.parafac`. Incompatible with the `like` argument.

`like` A `feemparafac` object.

In `reorder`, the factors in `x` will be reordered to match the factors in `like` according to the smallest of the cosine similarities (`congru`) for the emission and excitation wavelengths.

In `rescale`, every factor matrix $\mathbf{A}$ from `x` that was specified in `mode` will be multiplied by scaling factors $\mathbf{c}$ so in order to minimise $\|\mathbf{A}_{\texttt{like}} - \mathrm{diag}(\mathbf{c})\mathbf{A}_{\texttt{x}}\|^2$.

`mode` The modes to rescale, with `"A"`, `"B"`, `"C"` corresponding to emission, excitation, and samples, respectively. When `like` is specified, defaults to `c("A", "B")`. Forwarded to `rescale.parafac`.

`newscale` The desired root-mean-square for each column of the modes being rescaled. Forwarded to `rescale.parafac`. Incompatible with the `like` argument.

`absorb` The mode that should absorb the inverse rescaling coefficients. When `like` is specified, defaults to `"C"`. Forwarded to `rescale.parafac`.

Details

`feemparafac` 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.

After the PARAFAC decomposition is calculated, the scores are multiplied by the `scales` attribute of the `X` object, making them represent the cube with scaling undone. Use `feemscale(remember = FALSE)` if you don't want to undo the scaling.

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), simplify = FALSE)`

The `subset` and `envir` options are useful to repeatedly perform PARAFAC on different subsets of the same FEEM cube, e.g. in jack-knifing or split-half analysis. Since `feemparafac` keeps a reference to the its `X` and `envir` arguments, the use of `subset` should ensure that the same FEEM cube is referenced from multiple `feemparafac` objects instead of creating copies of its subsets. Additionally, environment objects are not duplicated on `save` or `load`, so storing `X` in an environment and passing it to multiple invocations of `feemparafac` will save a lot of memory when the results are serialised together.

`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 following attributes added:

cube A copy of the `X` argument.

subset A copy of the `subset` argument.

envir A copy of the `envir` argument.

time The time it took to compute the decomposition as an object of class `proc_time`. Only the time taken to run `multiway::parafac` (including the retries, if the decomposition failed) is included. The time taken to rescale the scores should be negligible.

`rownames` are added from the original data cube to the `A`, `B`, `C` components of the list returned by `parafac`.

Use `feemcube` on the return value to access the original data cube.

`plot.feemparafac`

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

`coef.feemparafac`

A `data.frame` or a list of them (only if `type` is "all"). See the description of the `type` argument for more information.

`fitted.feemparafac`

A `feemcube` object comparable to `X` as it was decomposed by `parafac`, ignoring the scaling.

`resid.feemparafac`

A `feemcube` object equal to $\mathbf{X} - \hat{\mathbf{X}}$, with an additional class `feem.resid` set. Objects of this class are plotted with a different default palette, see `plot.feem.resid`.

References

Bro R (1997). "PARAFAC. Tutorial and applications." *Chemometrics and Intelligent Laboratory Systems*, **38**(2), 149-171. doi:10.1016/S01697439(97)000324.

See Also

The `parafac` class structure; `write.openfluor`, `feemcube` for methods specific to values returned from this function.

The `rescale` generic is re-exported from the **multiway** package.

Examples

```
data(feems)
cube <- feemscale(feemscatter(cube, c(24, 14)), na.rm = TRUE)
(factors <- feemparafac(cube, nfac = 3, ctol = 1e-4))
plot(factors, 'image')
plot(factors, 'line')
head(coef(factors, 'loadings'))
str(coef(factors, 'all'))
str(feemcube(factors)) # original cube is retained
plot(fitted(factors))
plot(resid(factors))
```

feems — *Synthetic fluorescence excitation-emission matrices and absorbance spectra*

Description

This dataset consists of twelve fluorescence and absorbance spectra simulated from three trilinear components, with scattering signal added and divided by a correction factor to simulate inner filter effect.

Usage

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

Format

feems A named list of 12 `feem` objects containing fluorescence data measured with excitation wavelengths between 230 nm and 350 nm (with a step of 2 nm) and emission wavelengths between 240 nm and 435 (with a step of 5 nm).

cube A 12-sample `feemcube` object consisting of of 32 by 10 FEEMs measured at the same wavelength range as above with inner filter effect corrected.

absorp A 12-element named list containing absorbance spectra measured between 230 and 450 nm in 1 cm cells. Each element of the list is a two-column matrix. The first column contains the wavelengths and the second column contains the absorbance values.

Examples

```
data(feems)
plot(cube)
plot(feems$a)
matplot(
  absorp[[1]][,1],
  sapply(absorp, function(x) x[,2]),
  type = 'l', lty = 1
)
```

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, ..., progress = FALSE)
  ## S3 method for class 'list'
feemscale(x, ..., progress = FALSE)
```

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, to feemscale.feem, then to the norm function. Use this to set na.rm = TRUE for functions like sd or sumsq.
progress	Set to TRUE to enable a progress bar (implemented via txtProgressBar).

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.

References

Bro R, Smilde AK (2003). "Centering and scaling in component analysis." *Journal of Chemometrics*, **17**(1), 16-33. doi:10.1002/cem.773.

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, ..., cl, progress = TRUE)
  ## S3 method for class 'feemcube'
feemscatter(x, ..., cl, progress = TRUE)
  ## S3 method for class 'feem'
feemscatter(
    x, widths, method = c("omit", "pchip", "loess", "kriging", "whittaker"),
    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 or a list containing the half-widths of the scattering bands, in nm. Rayleigh scattering is followed by Raman scattering, followed by second diffraction order for Rayleigh and Raman, and so on. (Typically, there's no need for anything higher than third order, and even that is rare.) For example:

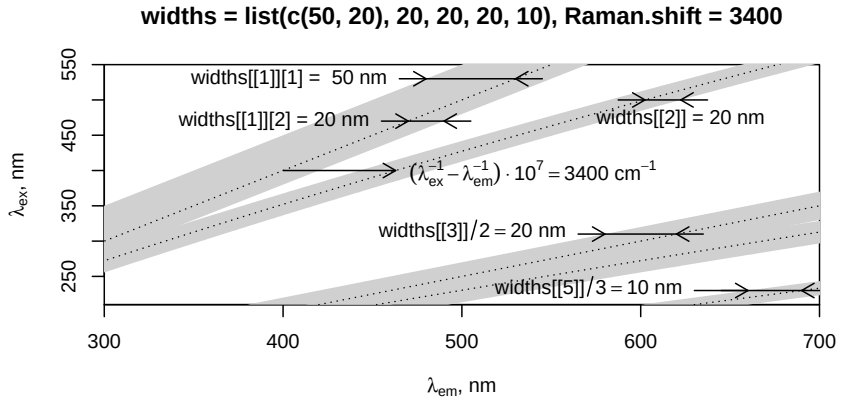
1. Rayleigh scattering
2. Raman scattering
3. Rayleigh scattering, 2λ
4. Raman scattering, 2λ
5. ...

For higher diffraction orders, the peak widths are proportionally scaled, making it possible to provide the same number for all kinds of scattering visible in the EEM. Set a width to 0 if you don't want to handle this particular kind of scattering signal.

It's possible to specify the bands asymmetrically. If the area to be corrected should range from x nm to the left of the scattering peak to y nm to the right of it, pass a list instead of a vector, and put a two-element vector `c(x, y)` for the appropriate kind of scattering. For example, passing `widths = list(c(30, 20), 20)` means "handle -30 nm to the left and $+20$ nm to the right of Rayleigh peak and ± 20 nm around Raman peak".

To sum up, given two half-widths W_1 and W_2, the test for being inside a kth diffraction order of a scattering band is as follows:

$$-W_1 < \frac{\lambda_{\text{center}}}{k} - \lambda_{\text{em}} < +W_2$$



In this example, a much larger portion of the anti-Stokes area is removed near the first order Rayleigh scattering signal than in the Stokes area. This can be useful to get rid of undesired signal where no fluorescence is observed on some spectrometers. The second and third order scattering signal areas are automatically scaled 2 and 3 times, respectively.

`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`). Pass a `by` argument to choose the direction of interpolation; see Details.

loess Interpolate it by fitting a locally weighted polynomial surface (`loess`). Extra arguments are passed verbatim to `loess`, which may be used to set parameters such as `span`.

kriging Interpolate it by means of ordinary or simple Kriging, as implemented in **pracma** function `kriging`. Pass a `type` argument to choose between the two methods. This method is not recommended due to its high CPU time and memory demands: it has to invert a dense $O(N^2)$ matrix (which easily reaches multiple gigabytes for some EEMs), and compute its product with a vector then take scalar products $O(N)$ times, with $N =$ `length(x)`.

whittaker Interpolate it by minimising a weighted sum of squared residuals (for known part of the spectrum) and roughness penalty (squared central difference approximations for derivatives by λ_{em} and λ_{ex}). See Details for more information and parameters.

`add.zeroes` Set intensities at $\lambda_{\text{em}} < \lambda_{\text{ex}} -$ `add.zeroes` nm to 0 unless they have been measured in order to stabilise the resulting decomposition (Thygesen, Rinnan, Barsberg, and Møller 2004). 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 "pchip" method is selected, the `by` parameter chooses between interpolating by row, by column, or averaging both, see Details. If "loess" method is selected, remaining options are passed to `loess` (the `span` parameter is of particular interest there). If "kriging" method is selected, remaining options are passed to `kriging`. If "whittaker" method is selected, available parameters include `d`, `lambda`, `nonneg` and `logscale`, see Details.
`cl`	If not `missing`, a **parallel** cluster object to run the scattering correction code on or `NULL` for the default cluster object registered via `setDefaultCluster`.
`progress`	Set to `FALSE` to disable the progress bar.

Details

The "pchip" method works by default as described in (Bahram, Bro, Stedmon, and Afkhami 2006): each emission spectrum at different excitation wavelengths is considered one by one. Zeroes are inserted in the corners of the spectrum if they are undefined (`NA`) to prevent extrapolation from blowing up, then the margins are interpolated using the corner points, then the rest of the spectrum is interpolated line by line. Since `pchip` requires at least 3 points to interpolate, the function falls back to linear interpolation if it has only two defined points to work with. The `by` argument controls whether the function proceeds by rows of the matrix ("emission", default), by columns of the matrix ("excitation"), or does both ("both") and averages the results to make the resulting artefacts less severe (Pucher, Wünsch, Weigelhofer, Murphy, Hein, and Graeber 2019) (see the **staRdom** package itself).

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 spectrum. Any negative values predicted by `loess` are replaced by 0.

The "kriging" method (Press, Teukolsky, Vetterling, and Flannery 2007) is much more computationally expensive than the previous two, but, on some spectra, provides best results, not affected by artefacts resulting from line-by-line one-dimensional interpolation (`pchip`) or varying degrees of smoothness in different areas of the spectrum (`loess`). Any negative values returned by `kriging` are replaced by 0.

Whittaker smoothing: The "whittaker" method (Krylov and Labutin 2023) works by minimising a sum of penalties, requiring the interpolated surface to be close to the original points around it and to be smooth in terms of derivatives by λ_{em} and λ_{ex}.

The parameters `d` and `lambda` should be numeric vectors of the same length, corresponding to the derivative orders (whole numbers ≥ 1) and their respective penalty weights (small real numbers; larger is smoother). For interpolation purposes, the default penalty is $10^{-2}\mathbf{D}_1 + 10\mathbf{D}_2$, which corresponds to `d = 1:2` and `lambda = c(1e-2, 10)`.

Any resulting negative values are pulled towards 0 by adding zero-valued points with weight `nonneg` (default 1) and retrying. Set `nonneg` to 0 to disable this behaviour. It is also possible to deal with resulting negative values by scaling and shifting the signal between `logscale` (typically) and 1, interpolating the logarithm of the signal, then undoing the transformation. By default `logscale` is `NA`, disabling this behaviour, since it may negatively affect the shape of interpolated signal.

See the internal help page whittaker2 for implementation details.

Value

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

References

Bahram M, Bro R, Stedmon C, Afkhami A (2006). "Handling of Rayleigh and Raman scatter for PARAFAC modeling of fluorescence data using interpolation." *Journal of Chemometrics*, **20**(3-4), 99-105. doi:10.1002/cem.978.

Krylov IN, Labutin TA (2023). "Recovering fluorescence spectra hidden by scattering signal: in search of the best smoother." *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 122441. doi:10.1016/j.saa.2023.122441.

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007). "Interpolation by Kriging." In *Numerical recipes: The Art of Scientific Computing (3rd Ed.)*, chapter 3.7.4, 144-147. Cambridge University Press, New York.

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

Thygesen LG, Rinnan Å, Barsberg S, Møller JKS (2004). "Stabilizing the PARAFAC decomposition of fluorescence spectra by insertion of zeros outside the data area." *Chemometrics and Intelligent Laboratory Systems*, **71**(2), 97-106. ISSN 0169-7439, doi:10.1016/j.chemolab.2003.12.012.

See Also

`feem`, `feemcube`

Examples

```
data(feems)
plot(x <- feemscatter(
  feems[[1]], widths = c(25, 25, 20, 20),
  method = 'whittaker', Raman.shift = 3500
))
```

`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 (DeSarbo 1984).

Usage

```
  feemsplithalf(
    cube, nfac, splits, random, groups, fixed, ..., progress = TRUE
  )
  ## S3 method for class 'feemsplithalf'
plot(
    x, kind = c('tcc', 'factors', 'aggtcc', 'bandfactors'), ...
  )
  ## S3 method for class 'feemsplithalf'
print(x, ...)
  ## S3 method for class 'feemsplithalf'
coef(
    object, kind = c('tcc', 'factors', 'aggtcc', 'bandfactors'), ...
  )
```

Arguments

cube	A feemcube object.
nfac	An integer vector of numbers of factors to check.
splits	A scalar or a two-element vector consisting of whole numbers. The first element is the number of parts to split the data cube into, which must be even. After splitting, the parts are recombined into non-intersecting halves (Murphy, Stedmon, Graeber, and Bro 2013), which are subjected to PARAFAC decomposition and compared against each other. The second element, if specified, limits the total number of comparisons between the pairs, since the number of potential ways to recombine the parts of the data cube into halves grows very quickly. The number of PARAFAC models fitted is $2 \cdot$ splits[2]. If only splits[1] is specified, splits[2] defaults to $\binom{\texttt{splits[1]}}{\texttt{splits[1]}/2}$. Mutually incompatible with the parameters random, fixed.
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 (Krylov, Drozdova, and Labutin 2020). The number of PARAFAC models fitted is $2 \cdot$ random. Mutually incompatible with the parameters splits, fixed.
groups	Use this argument to preserve the ratios between the groups present in the original dataset when constructing the halves. If specified, must be a factor or an integer vector of length dim(cube)[3] (specifying the group each sample belongs to) or a list of them, i.e., a valid f argument to split. By default, samples are considered to form a single group. For the split-combine method (splits), each group must have at least splits elements; for best results, sizes of groups should be close to a multiple of splits. For the randomised split-half method (random), each group should have at least 2 elements. Mutually incompatible with the fixed parameter.

`fixed` Use this argument to manually specify the contents of the halves to test. The argument must be a list containing two-element lists specifying the halves to compare. Each half must be a vector consisting of whole numbers specifying sample indices in the cube (see the example).

It is considered an error to specify a sample in both halves.

Mutually incompatible with the parameters `splits`, `random`, `groups`.

`progress` Set to FALSE to disable the progress bar.

`x`, `object` An object returned by `feemsplithalf`.

`kind` Chooses what type of data to return or plot:

tcc 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.
When plotting, TCC values for the component with the same number have the same colour.

factors The resulting loading values.
When plotting, split the plot into panels per each number of components and each mode (emission or excitation). Components with the same number have the same colour.

aggtcc `aggregate` the TCCs returned by `coef(x, 'tcc')` over individual components. By default, the function returns the minimal values, but a different aggregation can be chosen using the additional argument `FUN`.
When plotting, use a combination of a box-and-whiskers plot and a violin plot.

bandfactors `aggregate` the factor values from `coef(x, 'factors')` over individual tests and halves. By default, collect the 2.5% and 97.5% quantiles as the lower and the upper boundaries, respectively, and medians as the estimates.
The additional argument `FUN` can be specified as a function returning `c(lower, estimate, upper)` given a numeric vector. The additional argument `subset` works as in `base::subset`.
When plotting, the estimates are lines on top of semi-transparent polygons signifying the lower and upper boundaries..

`...` **feemsplithalf** Remaining options are passed to `feemparafac` and, eventually, to `parafac`. It is recommended to fully name the parameters instead of relying on partial or positional matching.
The arguments `parallel`, `cl` are handled separately, see Details.

plot.feemsplithalf Passed to `xyplot`.

print.feemsplithalf No additional options are allowed.

coef.feemsplithalf Ignored unless `kind %in% c('aggtcc', 'bandfactors')`.

Details

As the models (loadings $\mathbf{A}$, $\mathbf{B}$ and scores $\mathbf{C}$) 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 (Riu

and Bro 2003) 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 $\mathrm{diag}(\mathbf{s}_\mathrm{A}) \times \mathrm{diag}(\mathbf{s}_\mathrm{B}) \times \mathrm{diag}(\mathbf{s}_\mathrm{C}) = \mathbf{I}$, to make them comparable.

To perform stratified sampling on a real-valued variable (e.g. salinity, depth), consider binning samples into groups using `cut`, perhaps after histogram flattening using `ecdf(x)(x)`. To determine the number of breaks, consider `nclass.Sturges`.

To conserve memory, `feemsplithalf` puts the user-provided `cube` in an environment and passes it via `envir` and `subset` options of `feemparafac`. This means that, unlike in `feemparafac`, the `cube` argument has to be a `feemcube` object and passing `envir` and `subset` options to `feemsplithalf` is not supported.

Instead of forwarding the arguments `parallel`, `cl` to `multiway::parafac`, `feemsplithalf` schedules the calls to `feemparafac` on the cluster by itself. This makes it possible to fit more than `nstart` models at the same time if enough nodes are present in the **parallel** cluster `cl`.

`plot.feemsplithalf` plots results of the split-half procedure (TCC or loading values depending on the `kind` argument) using **lattice** graphics. Sane defaults are provided for `xyplot` parameters `xlab`, `ylab`, `as.table`, but they can be overridden.

`print.feemsplithalf` displays a very short summary of the analysis, currently the minimum TCC value for each number of components.

`coef.feemsplithalf` returns the Tucker's congruence coefficients resulting from the split-half analysis.

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[i]`), 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.

coef.feemsplithalf A `data.frame` containing various columns, depending on the value of the `kind` argument:

- **tcc** **factor** The factor (out of `nfac`) under consideration.
- **tcc** Tucker's congruence coefficient between a pair of matching components. Out of two possible values (TCC between excitation loadings or emission loadings), the minimal one is chosen, because the same rule is used to find which components match when reordering them in a pair of models.
- **test** The sequence number for each pair of models in the split-half test, related to the third dimension of `object$factors` or `object$tcc`. May be used to group values for plotting or aggregation.

subset Consists of two-element lists containing indices of the samples in each half of the original cube.

nfac The number of factors in the pair of models under consideration.

factors **wavelength** Emission and excitation wavelengths.

value The values of the loadings.

factor Number of the factor, 1 to `nfac`.

mode The mode the loading value belongs to, "Emission" or "Excitation".

nfac Total number of factors.

test Sequence number of a split-half test, indicating a given way to split the dataset in a group of splits with the same numbers of factors.

half Number of the half, 1 or 2.

subset For every row, this is an integer vector indicating the subset of the original data cube that the loadings have been obtained from.

aggtcc The columns `tcc`, `nfac`, `test` after aggregation of `coef(kind = 'tcc')`.

bandfactors Columns `wavelength`, `factor`, `mode`, `nfac` from `coef(kind = 'factors')`, plus columns `lower`, `estimate`, `upper` signifying the outputs from the aggregation function.

References

DeSarbo WS (1984). "An Application of PARAFAC to a Small Sample Problem, Demonstrating Preprocessing, Orthogonality Constraints, and Split-Half Diagnostic Techniques (Appendix)." *Research Methods for Multimode Data Analysis*, 602-642. https://papers.ssrn.com/abstract=2783446.

Krylov I, Drozdova A, Labutin T (2020). "Albatross R package to study PARAFAC components of DOM fluorescence from mixing zones of arctic shelf seas." *Chemometrics and Intelligent Laboratory Systems*, **207**(104176). doi:10.1016/j.chemolab.2020.104176.

Murphy KR, Stedmon CA, Graeber D, Bro R (2013). "Fluorescence spectroscopy and multi-way techniques. PARAFAC." *Analytical Methods*, **5**, 6557-6566. doi:10.1039/c3ay41160e.

Riu J, Bro R (2003). "Jack-knife technique for outlier detection and estimation of standard errors in PARAFAC models." *Chemometrics and Intelligent Laboratory Systems*, **65**(1), 35-49. doi:10.1016/S01697439(02)000904.

See Also

`feemparafac`, `parafac`, `congru`, `feemcube`.

Examples

```
data(feems)
cube <- feemscale(feemscatter(cube, rep(14, 4)), na.rm = TRUE)

(sh <- feemsplithalf(
  cube, 1:4, splits = 4, # => S4C6T3
  # splits = c(4, 2) would be S4C4T2, and so on
  # the rest is passed to multiway::parafac;
  ctol = 1e-4
```

```
  # here we set a mild stopping criterion for speed;
  # be sure to use a stricter one for real tasks
))

# specifying fixed halves to compare as list of 2-element lists
fixed <- list(
  list(1:6, 7:12),
  list(seq(1, 11, 2), seq(2, 12, 2))
)
sh.f <- feemsplithalf(cube, 2:3, fixed = fixed, ctol = 1e-4)

plot(sh, 'aggtcc')
head(coef(sh, 'factors'))
```

`marine.colours` *Perceptually uniform palettes*

Description

Create perceptually continuous palettes of R colours.

Usage

```
marine.colours(
  n, chroma = 0.65, luminance = c(0.35, 1),
  alpha = 1, gamma = 1, fixup = TRUE
)
diverging.colours(
  n, chroma = c(.1, .75), luminance = c(1, .35),
  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/alpha growth formulae. May be useful when it is needed to sacrifice the perceptual linearity of the palette to provide more contrast 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 NAs, but fixing the palette may make it less perceptually uniform.

Details

The marine.colours palette is used by default by all plot methods (e.g. plot.feem) for FEEM-like data to show absolute values. It is designed to retain perceptual uniformity even after complete desaturation.

The diverging.colours palette is used by plot.feem.resid to display residual values. People with severe colour vision deficiency (tritanopia or monochromacy) won't be able to discern positive and negative branches of the palette, but it's supposed to be legible for people with deuteranopia and protanopia.

Value

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

The marine.colours palette at the default values of $C^*_{uv} = 0.65$, $L^* \in [0.35; 1]$, $\alpha = \gamma = 1$.

The diverging.colours palette at the default values of $C^*_{uv} \in [0.1; 0.75]$, $L^* \in [0.35; 1]$, $\alpha = \gamma = 1$.

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.

CUBEHELIX (https://people.phy.cam.ac.uk/dag9/CUBEHELIX/) is a similar technique using BT.601 luminance coefficients and RGB colour space.

See Also

plot.feem, hcl

Examples

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

`plot.feem` *Plot a FEEM object*

Description

Plot a 2D fluorescence intensity surface as a pseudo-colour image.

Usage

```
## S3 method for class 'feem'
plot(
  x, xlab, ylab, cuts = 128,
  col.regions = marine.colours(256), ...
)
## S3 method for class 'feemcube'
plot(
  x, xlab, ylab, cuts = 128,
  col.regions = marine.colours(256), as.table = TRUE, ...
)
## S3 method for class 'feem.resid'
plot(
  x, ..., at, col.regions = diverging.colours(256)
)
```

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 assigned different colours.
`col.regions`	The palette to take the colours from, a character vector of R colour specifications.
`at`	The breakpoints in the intensity values. For residual plots, automatically set in a symmetric manner, ignoring the `cuts` argument.
`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))
```

`write.openfluor` *Export a PARAFAC model for the OpenFluor database*

Description

Prepares a fitted PARAFAC model for submission to OpenFluor - an online spectral database of fluorescence by environmental organic compounds.

Usage

```
write.openfluor(
  model, filename, name = "?", creator = "?", doi = "?",
  reference = "?", unit = "?", toolbox =, date =, fluorometer = "?",
  constraints =, validation = "?", methods = "?", preprocess = "?",
  sources = "?", ecozones = "?", description = "",
  shift = FALSE, scale = TRUE
)
```

Arguments

`model`	A `feemparafac` object.
`filename`	Path to the text file to create from the `model` argument.
`name`	Short name of the model.
`creator`	Name of the creator of the model.
`doi`	Digital object identifier of the referenced source. Can also be "ISBN:..." for books.
`reference`	Full citation for the referenced source using the following style: "Author AA, Author BB, Author CC, (year), 'Title', Journal Abbrev, Vol, pages".
`unit`	Units the fluorescence was measured in. Typically, one of "RU", "QSE", "AU".
`toolbox`	Defaults to "albatross *version*, multiway *version*".
`date`	Defaults to today, in "*yyyy-mm-dd*" format.
`fluorometer`	The model of the instrument that produced the data.
`constraints`	Constraints applied to the PARAFAC model. Defaults to `model$const`, but please edit it to a more human-readable form.
`validation`	Validation method used for the PARAFAC model, examples include: "Split-Half Analysis", "core-consistency".
`methods`	The sequence of steps taken to handle the samples and to ensure proper fluorescence intensity measurement. Examples include: • Sampling: Filtration GF/F

- Sampling: Filtration *x* um
- Sampling: samples frozen
- Instrument spectral bias correction: Ex
- Instrument spectral bias correction: Em
- Instrument spectral bias correction: Ex & Em
- Inner filter effect correction: absorbance method
- Inner filter effect correction: dilution
- Inner filter effect correction: CDA
- Inner filter effect correction: *other (please describe)*
- Internal calibration: Raman Peak area
- Internal calibration: Raman Peak height
- Internal calibration: Blank Subtraction
- External calibration: Quinine Sulphate dilution series
- External calibration: STARNA reference standards
- External calibration: NIST reference standards
- External calibration: *other (please describe)*

`preprocess` PARAFAC-specific pre-processing steps applied to the dataset. Examples include (but are not limited to):

- Outliers removed
- Scatter region excised (replaced with NaNs)
- Scatter region smoothed (replaced with interpolated values)
- Sample mode normalised to DOC concentration
- Sample mode normalised to unit variance

`sources` Should preferably include one or more of the following keywords:

- river
- stream
- lake
- wetland
- reservoir
- estuary
- ocean - coastal and shelf seas
- ocean - surface off-shore
- ocean - deep off-shore
- freshwater
- seawater
- groundwater
- wastewater
- drinking water
- treated water
- recycled water
- ballast water
- sediment

- mudflat
- mangrove
- aquarium
- mesocosm

`ecozones` List all major or minor terrestrial, freshwater and marine ecozones and ecoregions that apply. The full set of possible options is too large to include here, but see https://en.wikipedia.org/wiki/Lists_of_ecoregions for a source of inspiration.

`description` Brief description of the model and its source data in ≤ 256 characters.

`shift`, `scale` If `shift` is specified (default `FALSE`), the loadings are first shifted by subtracting `min(x)` to ensure that the minimal value is 0.

If `scale` is specified (default `TRUE`), the loadings are then rescaled by dividing by `max(x)` so that the maximal value is 1.

Note that OpenFluor clamps values outside the $[0, 1]$ range and uses scale-invariant (but *not* shift-invariant) Tucker's congruence coefficient to find matches.

Details

Provided the `model` and the `filename` arguments, this function exports the loadings into a file that passes OpenFluor syntax check and is suitable for further editing. Alternatively, some or all of the fields may be specified programmatically.

The fields `constraints`, `methods`, `preprocess`, `sources`, `ecozones` can be specified as character vectors (to be comma-separated on output); others should be single strings.

References

Murphy KR, Stedmon CA, Wenig P, Bro R (2014). "OpenFluor - an online spectral library of auto-fluorescence by organic compounds in the environment." *Analytical Methods*, **6**, 658-661. doi:10.1039/C3AY41935E.

https://openfluor.lablicate.com/

See Also

`feemparafac`

Examples

```
data(feems)
cube <- feemscale(feemscatter(cube, c(24, 14)), na.rm = TRUE)
factors <- feemparafac(cube, nfac = 3, ctol = 1e-4)
# all defaults
write.openfluor(factors, f1 <- tempfile(fileext = '.txt'))
if (interactive()) file.show(f1)
unlink(f1)
# all non-default arguments
write.openfluor(
  factors, f2 <- tempfile(fileext = '.txt'), name = 'example',
  creator = 'J. Doe', doi = '10.1000/1', reference = paste(
```

```
    'Upper D, (1973),', "'The unsuccessful self-treatment of a case",
    "of \"writer's block\"'," , 'J Appl Behav Anal, 7(3), 497'
  ), unit = 'AU', toolbox = 'all calculations done by hand',
  date = '2038-01-19', fluorometer = 'Acme Fluor-o-matic 9000',
  constraints = 'non-negative', validation = 'prior knowledge',
  methods = 'Instrument spectral bias correction: Ex & Em',
  preprocess = 'Scatter region excised (replaced with NaNs)',
  sources = 'freshwater', ecozones = 'Balkash',
  description = 'not a real model', shift = FALSE, scale = TRUE
)
if (interactive()) file.show(f2)
unlink(f2)
```

`[.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 wavelengths 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 (see Extract), may be integer, logical or character vectors, or missing.
`drop`	Coerce result to the lowest possible dimension (dropping the `feem` class if so).
`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. Use vector subsetting (zero or one argument inside the brackets) to disable the check.

Value

For `[`: If `drop` is `TRUE` 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).
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. Use vector subsetting (zero or one argument inside the brackets) to disable the check.

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