Package ‘albatross’

July 14, 2021

Type Package

Title PARAFAC Analysis of Fluorescence Excitation-Emission Matrices

Version 0.3-2

Depends R (>= 3.3)

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

Enhances eemR, EEM

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

License GPL (>= 3)

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2021-07-14 20:10:02 UTC

R topics documented:

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

Description

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

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

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

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

Details

Index of help topics:

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

`as.list.feemcube`	`Transform a FEEM cube object into a list of FEEM objects`
`feem`	`Create a fluorescence excitation-emission matrix object`
`feemcube`	`Build a data cube of FEEMs`
`feemgrid`	`Interpolate FEEMs on a given wavelength grid`
`feemife`	`Absorbance-based inner filter effect correction`
`feemjackknife`	`Jack-knife outlier detection in PARAFAC models`
`feemlist`	`Create lists of FEEM objects`
`feemparafac`	`Compute PARAFAC on a FEEM cube object`
`feems`	`Fluorescence excitation-emission matrices and absorbance spectra`
`feemscale`	`Rescale FEEM spectra to a given norm and remember the scale factor`
`feemscatter`	`Handle scattering signal in FEEMs`
`feemsplithalf`	`Split-half analysis of PARAFAC models`
`fitted.feemparafac`	`Extract fitted PARAFAC values or residuals`
`marine.colours`	`Marine colours`
`plot.feem`	`Plot a FEEM object`
`write.openfluor`	`Export a PARAFAC model for the OpenFluor database`

Author(s)

Timur Labutin

Maintainer: Ivan Krylov

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.

See Also

`feem`, `feemlist`, `feemife`, `feemscatter`, `feemgrid`, `feemcube`, `feemscale`, `feemsplithalf`, `feemparafac`, `feemjackknife`.

Examples

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

data(feems)

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


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


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

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

Description

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

Usage

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

Arguments

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

Details

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

Value

A `data.frame` containing three numeric columns:

`emission`	Emission wavelength, nm.
`excitation`	Excitation 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))
```

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

Description

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

Usage

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

Arguments

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

Value

A named list of FEEM objects comprising `x`.

See Also

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

Examples

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

`feem` *Create a fluorescence excitation-emission matrix object*

Description

Functions to create fluorescence excitation-emission matrix objects from R matrices coupled with excitation and emission wavelengths, 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 conversing a matrix, this should be a vector of emission wavelengths, each wavelength corresponding to a row of the matrix. If converting a `data.frame`, this optional argument specifies the name of the column containing the emission wavelengths.
`excitation`	If conversing a matrix, this should be a vector of excitation wavelengths, each wavelength corresponding to a column of the matrix. If converting a `data.frame`, this optional argument specifies the name of the column containing the excitation wavelengths.

`intensity`	If converting a `data.frame`, this optional argument specifies the name of the column containing the fluorescence intensities.
`scale`	The scale value of a EEM is preserved through the analysis procedure to divide the resulting score values after running PARAFAC. If the EEM has been premultiplied 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) but they are hard to distinguish from measured values (e.g. stored as zeroes instead of `NA`), 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 $\lambda_{\mathrm{em}} - \lambda_{\mathrm{ex}} < \Delta$, $X(\lambda_{\mathrm{em}}, \lambda_{\mathrm{ex}}) \in \mathtt{na}$) 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). The file contents are decoded into ASCII, dropping the unrepresentable bytes (since we only care about numbers), before being passed to `read.table`. This only works when `file` is a file path, not a connection.

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.

`...` 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.

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 = ','
)
```

---

feemcube *Build a data cube of FEEMs*

Description

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

Usage

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

Arguments

x	A list of FEEM objects, possibly named, or a numeric array. Alternatively, a feemparafac 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` returns the original data analysed by `feemparafac`.

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.

See Also

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

Examples

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

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 `NA`s that haven't been measured and `NA`s 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.

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. https://www.springer.com/la/book/9780387312781.

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)

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

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

Description

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

Usage

```
  feemjackknife(cube, ..., 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 `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.

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:

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

  - **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`

Examples

```
data(feems)
cube <- feemscale(
  feemscatter(feemcube(feems, FALSE), rep(24, 4))[1:30*6, 1:9*6,],
  na.rm = TRUE
)
# takes a long time
jk <- feemjackknife(cube, nfac = 3, const = rep('nonneg', 3))
# 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, ...
  )
  ## 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 **eemR** package or an `EEM` object from **EEM** package.
`format`	Corresponds to the `format` argument of `feem`. Currently, one format is assumed for all files to be imported.
`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.
`...`	When importing files, remaining options are passed to `feem`. 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 this mechanism behaving exactly as specified as it may be changed in the future versions.

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')
))
```

feemparafac — *Compute PARAFAC on a FEEM cube object*

Description

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

Usage

```
  feemparafac(
    X, ..., rescale = 3, retries = 10, 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", "emission", "excitation", "samples"
    ), ...
  )
```

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`. **coef.feemparafac** No other parameters are allowed.
`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.

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.

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.

`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.

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

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

Examples

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

feems *Fluorescence excitation-emission matrices and absorbance spectra*

Description

This dataset contains:

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

Usage

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

Format

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

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

Examples

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

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

Description

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

Usage

```
  feemscale(x, ...)
  ## S3 method for class 'feem'
feemscale(x, norm = sd, remember = TRUE, ...)
  ## S3 method for class 'feemcube'
feemscale(x, ..., 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 of length 4 containing the widths (in nm) of the scattering signal, in the following order: 1. Rayleigh scattering 2. Raman scattering 3. Rayleigh scattering, 2λ 4. Raman scattering, 2λ Set a width to 0 if you don't want to handle this particular kind of scattering signal.
`method`	A string choosing *how* to handle the scattering signal: **omit** Replace it with `NA`. **pchip** Interpolate it line-by-line using piecewise cubic Hermitean polynomials (`pchip`). Pass a `by` argument to choose the direction of interpolation; see Details. **loess** Interpolate it by fitting a locally weighted polynomial surface (`loess`). In this case the remaining parameters are passed verbatim to `loess`, which may be used to set parameters such as `span`. **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), then take its products with vectors $O(N)$ times. **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 $\lambda_{\mathrm{em}} < \lambda_{\mathrm{ex}} - \texttt{add.zeroes}\ \mathrm{nm}$ to 0 unless they have been measured. Set to `NA` to disable this behaviour.
`Raman.shift`	Raman shift of the scattering signal of water, cm^{-1}.

`...`	Passed verbatim from `feemscatter` generics to `feemscatter.feem`. If "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 [1]: 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 [2, 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 [3] 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.

The "whittaker" method [4] works by unfolding `x` into a vector $\mathbf{z}$ and looking for a vector $\hat{\mathbf{z}}$ that is close enough to $\mathbf{z}$, but also smooth, by minimising a sum of penalties:

$$(\mathbf{z}-\hat{\mathbf{z}})^\top \operatorname{diag}(\mathbf{w})(\mathbf{z}-\hat{\mathbf{z}}) + \sum_k \lambda_k |\mathbf{D}_{n_k}\hat{\mathbf{z}}|^2$$

The weights $\mathbf{w}$ are set to 0 for missing (`NA`) points and for those to be interpolated and to 1 otherwise. The matrix $\mathbf{D}_n$ is constructed in such a way that multiplying it by $\hat{\mathbf{z}}$ results in a vector of n-th order derivative estimates in both directions and in all applicable points of $\hat{\mathbf{z}}$ as a matrix. The wavelength grid is correctly taken into account by solving a Vandermonde system for every $n+1$ consecutive points.

The parameters `d` and `lambda` should be numeric vectors of the same length, choosing the difference orders (n_k) and their weights (λ_k). It has been shown in [5] that a combination of first- and second-order penalty ($2\lambda\mathbf{D}_1 + \lambda^2\mathbf{D}_2$) results in non-negative impulse response, but the resulting peak shape may be sub-optimal. Instead, the default penalty is $10^{-2}\mathbf{D}_1 + 10\mathbf{D}_2$, and resulting negative values are pulled to 0 with weight `nonneg` (default 1, same as fidelity weight) by adding a penalty of `nonneg` $\cdot \sum_i \mathbf{1}_{\hat{z}_i<0}\, \hat{z}_i^2$ and retrying until no new penalty weights are added. 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 10^{-4}) and 1, interpolating the logarithm of the signal, then undoing the transformation. This prevents the resulting values from getting lower than $\min(x) - (\max(x) - \min(x))\frac{\texttt{logscale}}{1-\texttt{logscale}}$, which is approximately $-\texttt{logscale}\cdot\max(x)$ if `logscale` and $\min(x)$ are both close to 0. By default `logscale` is `NA`, disabling this behaviour, since it may negatively affect the shape of interpolated signal.

Value

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

References

1. 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.
2. 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.
3. 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.
4. Eilers PHC (2003). "A Perfect Smoother." *Analytical Chemistry*, **75**(14), 3631-3636. doi: 10.1021/ac034173t.
5. Eilers PHC, Goeman JJ (2004). "Enhancing scatterplots with smoothed densities." *Bioinformatics*, **20**(5), 623-628. doi: 10.1093/bioinformatics/btg454.

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 [1].

Usage

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

Arguments

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

... **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.

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

print.feemsplithalf, coef.feemsplithalf No additional options are allowed.

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 [3] by minimising $||\mathbf{A}\,\mathrm{diag}(\mathbf{s}_\mathrm{A}) - \mathbf{A}^\mathrm{orig}||^2$ and $||\mathbf{B}\,\mathrm{diag}(\mathbf{s}_\mathrm{B}) - \mathbf{B}^\mathrm{orig}||^2$ over $\mathbf{s}_\mathrm{A}$ and $\mathbf{s}_\mathrm{B}$, subject to $\mathrm{diag}(\mathbf{s}_\mathrm{A}) \times \mathrm{diag}(\mathbf{s}_\mathrm{B}) \times \mathrm{diag}(\mathbf{s}_\mathrm{C}) = \mathbf{I}$, 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.

`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.

References

1. 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`.
2. 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.
3. 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(feemcube(feems, FALSE), rep(24, 4))[1:30*6, 1:9*6,],
  na.rm = TRUE
)
(sh <- feemsplithalf( # takes a long time
  cube, 2:4, splits = 4, # 4 splits => S4C6T3
  # the rest is passed to multiway::parafac
```

```
    const = rep('nonneg', 3) # setting ctol and maxit is recommended
  ))
  # feemparafac methods should be able to use the environment and subset
  plot(sh$factors[[1]])
  plot(sh)
  plot(sh, 'factors')
  head(coef(sh))
  head(coef(sh, 'factors'))
```

fitted.feemparafac *Extract fitted PARAFAC values or residuals*

Description

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

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

Usage

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

Arguments

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

Details

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

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

Value

A FEEM cube object.

See Also

feemcube, fitted.parafac, resid.

Examples

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

`marine.colours` *Marine colours*

Description

Create a perceptually contiguous palette of R colours, using hues typically associated with natural waters.

Usage

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

Arguments

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

Value

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

References

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

See Also

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

Examples

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

`plot.feem` *Plot a FEEM object*

Description

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

Usage

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

Arguments

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

Value

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

See Also

`levelplot`

Examples

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

`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(
    feemcube(feems, FALSE)[1:36*5, 1:11*5,],
    rep(24, 4)), na.rm = TRUE
)
factors <- feemparafac(cube, nfac = 2, const = rep('nonneg', 3))
# 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.

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.

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