Package ‘EMMAgeo’

October 12, 2022

Type  Package
Title  End-Member Modelling of Grain-Size Data
Version  0.9.7
Date  2019-12-16
Author  Michael Dietze, Elisabeth Dietze
Maintainer  Michael Dietze <mdietze@gfz-potsdam.de>
Description  End-member modelling analysis of grain-size data is an approach
to unmix a data set's underlying distributions and their contribution to
the data set. EMMAgeo provides deterministic and robust protocols for
that purpose.
License  GPL-3
Encoding  UTF-8
Depends  R (>= 3.5.0)
Imports  GPArotation, limSolve, caTools, shiny, matrixStats
RoxygenNote  7.0.2
NeedsCompilation  no
Repository  CRAN
Date/Publication  2019-12-16 15:10:06 UTC

R topics documented:

EMMAgeo-package ........................................ 2
check.data .............................................. 3
click.limits ........................................... 4
convert.units .......................................... 5
create.EM ................................................ 6
EMMA ......................................................... 7
EMpot ...................................................... 10
EMrob ...................................................... 10
get.l ......................................................... 11
get.l.opt ................................................ 12
EMMAgeo-package

End-member modelling algorithm and supporting functions for unmixing grain-size distributions and further compositional data.

Description

EMMAgeo provides a set of functions for end-member modelling analysis (EMMA) of grain-size data and other cases of compositional data. EMMA describes a multivariate data set of m samples, each comprising n parameters (e.g. grain-size classes), as a linear combination of end-member loadings (the underlying distributions) and end-member scores (the contribution of each loading to each sample).

EMMA can be run in two principal ways, a deterministic and a robust, including modelling the uncertainties. The deterministic way can be accessed simply with the function EMMA(). For the robust way there are two protocols that need to be respected. There is a compact protocol, which is mainly automated but needs adjustments by the user, and there is an extended protocol, which allows access to all parameterisation steps of robust EMMA.

The package contains further auxiliary functions to check and prepare input data, test parameters and use a graphic user interface for deterministic EMMA. The package also contains an example data set, comprising measured grain-size distributions of real world sediment end-members.

Details

| Package:   | EMMAgeo        |
| Type:      | Package        |
| Version:   | 0.9.7          |
| Date:      | 2019-05-10     |
| License:   | GPL-3          |
Author(s)

Michael Dietze, Elisabeth Dietze

check.data

Check correctness and consistency of input data

Description

The input data matrix (X), number of end-members (q), weight transformation limits (l) and constant sum scaling parameter (c) are checked. This includes checking for absence of missing values, columns containing only zero-values and for numeric data type of all variables. A further check tests if l is below the maximum possible value, preventing numerical instability prior to factor rotation.

Usage

check.data(X, q, l, c, ...)

Arguments

X Numeric matrix, input data set with m samples (rows) and n variables (columns).
q Numeric scalar, number of end-members to be modelled.
l Numeric scalar or vector, weight transformation limit, i.e. quantile.
c Numeric scalar, constant sum scaling parameter, e.g. 1, 100, 1000.
... Further arguments passed to the function.

Value

Character vector, verbose test results.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

EMMA
Examples

```r
## load example data set
data(example_X)

## perform data set check
check.data(X = X,
q = 6,
1 = seq(from = 0,
to = 0.2,
by = 0.01),
c = 1)
```

### click.limits

**Define mode limits by mouse clicks.**

#### Description

This function allows defining limits for robust end-members by mouse clicks on a combined plot output, showing a histogram and all end-members together. Clicks must be placed in the order lower limit, upper limit - for each end-member successively.

#### Usage

`click.limits(data, n, classunits)`

#### Arguments

- `data` List object, output of `test.robustness`.
- `n` Numeric scalar, number of target end-members.
- `classunits` Numeric vector, optional class units (e.g. micrometers or phi-units).

#### Value

Numeric matrix, limit classes. The first row contains lower limits, the second row upper limits for each end-member.

#### Author(s)

Michael Dietze, Elisabeth Dietze

#### See Also

test.robustness, robust.EM
convert.units

Examples

```r
## load example data set
data(example_X)

## Test robustness
q <- 4:7
l <- seq(from = 0, to = 0.1, by = 0.02)
TR <- test.robustness(X = X, q = q, l = l)

## define 2 limits by mouse clicks (uncomment to use).
# limits <- click.limits(data = TR, n = 2)
# limits
```

convert.units Convert between phi and micrometers.

Description

The function converts values from the phi-scale to the micrometer-scale and vice versa.

Usage

```r
convert.units(phi, mu)
```

Arguments

- `phi` Numeric vector, grain-size class values in phi to be converted.
- `mu` Numeric vector, grain-size class values in micrometres to be converted.

Value

Numeric vector, converted grain-size class values.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

interpolate.classes
**Examples**

```r
## generate phi-values
phi <- -2:5

## convert and show phi to mu
mu <- convert.units(phi = phi)
mu

## convert and show mu to phi
convert.units(mu = mu)
```

---

**create.EM**

*Create grain-size-distributions.*

**Description**

This function allows creating artificial grain-size end-members. One such "artificial end-member loading" may be composed of one or more superimposed normal distributions.

**Usage**

`create.EM(p1, p2, s, boundaries, n)`

**Arguments**

- `p1` `Numeric` vector, means of normal distributions, i.e. mode positions.
- `p2` `Numeric` vector, standard deviations of normal distributions, i.e. mode width.
- `s` `Numeric` vector, relative proportions of each mode, i.e. relative mode height.
- `boundaries` `Numeric` vector of length two with class boundaries (i.e. `c(lower boundary, upper boundary)`).
- `n` `Numeric` scalar with number of classes, i.e. resolution of the end-member.

**Details**

When building a data set of many artificial end member loadings, these should all have the same boundaries and `n`. The function builds composites of individual normal distributions. Each distribution is scaled according to `s`. Finally the distribution is scaled to 100 %.

**Value**

`Numeric` vector with normalised end-member loadings, consisting of the mixed normal distributions according to the input parameters.

**Author(s)**

Michael Dietze, Elisabeth Dietze
EMMA

See Also

mix.EM

Examples

```r
## set lower and upper class boundary, number of classes and class units
boundaries <- c(0, 11)
n <- 40
phi <- seq(from = boundaries[1],
           to = boundaries[2],
           length.out = n)

## create two artificial end-member loadings
EMa.1 <- create.EM(p1 = c(2, 5), p2 = c(1, 0.8), s = c(0.7, 0.3),
                   boundaries = boundaries, n = n)
EMa.2 <- create.EM(p1 = c(4, 7), p2 = c(1.1, 1.4), s = c(0.5, 0.5),
                   boundaries = boundaries, n = n)

## plot the two artificial end-member loadings
plot(phi, EMa.1, type = "l")
lines(phi, EMa.2, col = "red")
```

EMMA

End-member modelling analysis algorithm.

Description

A multivariate data set (m samples composed of n variables) is decomposed by eigenspace analysis and modelled with a given number of end-members (q). Several steps of scaling, transformation, normalisation, eigenspace decomposition, factor rotation, data modelling and evaluation are performed.

Usage

```r
EMMA(
  X,
  q,
  l,
  c,
  Vqn,
  classunits,
  ID,
  EM.ID,
  rotation = "Varimax",
  plot = FALSE,
  ...
)
```
**Arguments**

- **X**: Numeric matrix, input data set with m samples (rows) and n variables (columns).
- **q**: Numeric scalar, number of end-members to be modelled.
- **l**: Numeric scalar or vector, weight transformation limit, i.e. quantile. Set to zero if omitted.
- **c**: Numeric scalar, constant sum scaling parameter, e.g. 1, 100, 1000. Set to 100 if omitted.
- **Vqn**: Numeric matrix, optional unscaled user-defined end-member loadings. If provided, these are used instead of model-derived ones. See details.
- **classunits**: Numeric vector, optional class units (e.g. micrometers or phi-units) of the same length as columns of X.
- **ID**: Numeric or character vector, optional sample IDs of the same length as rows of X.
- **EM.ID**: Character vector, end-member names. If present, these will be set as row-names of the output data set and used in the legend text.
- **rotation**: Character scalar, rotation type, default is "Varimax". See details.
- **plot**: Logical scalar, optional graphical output of the results, default is FALSE. If set to TRUE, end-member loadings and end-member scores are plotted.
- **...**: Additional arguments passed to the plot function. Since the function returns two plots some additional graphical parameters must be specified as vector with the first element for the first plot and the second element for the second plot.

**Details**

The parameter Vqn is useful when EMMA shall be performed with a set of prior unscaled end-members, e.g. from other data sets that are to be used as reference or when modelling a data set with mean end-members, as in the output of robust.loadings.

The rotation type Varimax was used by Dietze et al. (2012). In this R package, one out of the rotations provided by the package GPArotation is possible, as well. However, tests showed that the rotation type has no dramatic consequences for the result.

The function values loadings and scores are redundant. They are essentially the same as Vqsn and Mqs. However, they are included for user convenience.

**Value**

A list with numeric matrix objects.

- **loadings**: Normalised rescaled end-member loadings.
- **scores**: Rescaled end-member scores.
- **Vqn**: Normalised end-member loadings.
- **Vqsn**: Normalised rescaled end-member loadings.
- **Mqs**: Rescaled end-member scores.
- **Xm**: Modelled data.
- **modes**: Mode class of end-member loadings.
EMMA

Mqs.var  Explained variance of end-members
Em  Absolute row-wise model error.
En  Absolute column-wise model error.
RMSEm  row-wise root mean square error
RMSEn  column-wise root mean square error
Rm  Row-wise (sample-wise) explained variance.
Rn  Column-wise (variable-wise) explained variance.
o1  Number of overlapping end-members.

Author(s)
Michael Dietze, Elisabeth Dietze

References

See Also
test.parameters, rotations, eigen, nnls

Examples

```r
## load example data and set phi-vector
data(example_X)
phi <- seq(from = 1, to = 10, length.out = ncol(X))

## perform EMMA with 5 end-members
EM <- EMMA(X = X, q = 5, l = 0.05, c = 100, plot = TRUE)

## perform EMMA with 4 end-members and more graphical settings
EM <- EMMA(X = X, q = 4, l = 0.05, c = 100, plot = TRUE,
            EM.ID = c("EM 1", "EM 2", "EM 3", "EM 4"),
            classunits = phi,
            xlab = c(expression(paste("Class [", phi, "]")), "Sample ID"),
            cex = 0.7,
            col = rainbow(n = 4))
```
### EMpot

**Description**

A list with output of the function test.robustness()

**Format**

The format is: List of 8 $q$: num [1:90] 4 4 4 4 4 4 4 4 4 ... $lw$: num [1:90] 0 0 0 0 0.05 0.05 0.05 0.05 0.1 0.1 ... $modes$: num [1:90] 12 32 61 80 12 32 61 80 12 32 ...

**Details**

The dataset is the result of the function test.robustness() of the R-package EMMAgeo.

**Examples**

```r
## load example data set
data(example_EMpot)
```

### EMrob

**Description**

Robust end-members, a list with output of the function robust.EM()

**Format**

The format is: List of 12 $Vqsn.data$:List of 4 ..$Vqsn.data$: num [1:15, 1:80] 0.18929 0.184 0.18304 0.00698 0.02033 ...

**Details**

The dataset is the result of the function robust.EM() of the R-package EMMAgeo.

**Examples**

```r
## load example data set
data(example_EMrob)
```
**get.l**

Generate a vector of weight transformation values from l.min to l.max.

**Description**

This function generates a sequence of weight transformation values that range from l_min (by default zero) to l_max (by default 95 % of the maximum possible value). It uses the function test.l.max().

**Usage**

get.l(X, n = 10, max = 0.95, min = 0)

**Arguments**

- **X** Numeric matrix, input data set with m samples (rows) and n variables (columns).
- **n** Numeric scalar, length of the output vector (by default 10).
- **max** Numeric scalar, fraction of the maximum value (by default 0.95).
- **min** Numeric scalar, minimum value (by default zero).

**Value**

Numeric vector of class \"EMMAGEO_l\", weight transformation values.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**See Also**

test.l.max

**Examples**

```r
## load example data set
data(example_X)

## infer l-vector
l <- get.l(X = X,
    n = 5,
    max = 0.8,
    min = 0.02)
```
get.l.opt

Identify optimum weight transformation value

Description

This function returns for a series of input values the weight transformation value, which yielded the highest measure of model quality.

Usage

get.l.opt(X, l, quality = "mRt", Vqn, rotation, plot = TRUE, ...)

Arguments

X Numeric matrix, input data set with m samples (rows) and n variables (columns).

l Numeric vector, weight transformation values to test.

quality Character scalar, quality measure for against which to test the influence of l. See details for a list of the available keywords. Default is "mRt".

Vqn Numeric matrix specifying optional unscaled user-defined end-member loadings.

rotation Character scalar, rotation type, default is "Varimax" (cf. EMMA for further information).

plot Logical scalar, optional graphical output of the result.

... Further arguments passed to the function.

Details

The parameter quality can be one out of the following keywords: "mRm", "mRn", "mRt", "mEm", "mEn" and "mEt". See EMMA for definition of these keywords.

Value

Numeric scalar, weight tranformation value with optimal EMMA result.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

EMMA
**get.limits**

Infer lower and upper mode position limits to define robust end-members.

### Description

This function identifies the lower and upper limits within which robust end-members have clustered mode positions. It uses a kernel density estimate of the mode positions of all input end-member loadings, clips it at a user-defined minimum density and returns the resulting rising and falling shoulders of the kde peaks as limits.

### Usage

```r
get.limits(loadings, classunits, bw, threshold = 0.7)
```

### Arguments

- **loadings**
  - Numeric matrix, m loadings (rows) and n classes (columns).
- **classunits**
  - Numeric vector, optional class units (e.g. micrometers or phi-units) of the same length as columns of `X`.
- **bw**
  - Numeric scalar, bandwidth of the kernel, moved over the data set. If omitted, the default value of 1 used.
- **threshold**
  - Numeric scalar, threshold quantile which is used to identify mode clusters. Only kde densities above this values are kept and used to derive mode cluster limits.

### Details

Note that the threshold above which a mode cluster is identified is an arbitrary, user-defined value and probably needs to be adjusted iteratively to get reasonable results. The default value may or may not be adequate!

### Value

Numeric matrix with lower and upper mode limits.
get.q

Author(s)
Michael Dietze, Elisabeth Dietze

See Also
EMMA, model.EM

Examples

```r
## load example data set
data(example_EMpot)

## infer mode cluster limits
limits <- get.limits(loadings = EMpot)
```

Description
This function uses the input data matrix X and a vector of weight transformation limits to generate a matrix of minimum and maximum likely numbers of end-members to be used to model and extract robust end-members.

Usage

```r
get.q(
  X,
  l = 0,
  q.min = 2,
  q.max = 10,
  criteria.min = 0.5,
  criteria.max = "local_max",
  correct.output = TRUE,
  ...
)
```

Arguments

- **X**: Numeric matrix, input data set with m samples (rows) and n variables (columns).
- **l**: Numeric vector, weight transformation limits, default is zero.
- **q.min**: Numeric scalar, minimum number of end-members to use, default is 2.
- **q.max**: Numeric scalar, maximum number of end-members to use, default is 10.
get.q

criteria.min Numeric scalar, minimum value of explained variance reached to be a valid model realisation, default is 0.5.
criteria.max Character or numeric scalar, either keyword "local_max" to use first local maximum or any numeric value of explained variance, default is "local_max".
correct.output Logical scalar, option to correct the output for twisted values and remove combinations with NA-values. See details.

Details

The parameter q.min should be at least 2 because otherwise the entire dataset would consist of one end-member and there would be no variability at all. The parameter q.max is set to 10 by default, based on practical issues. In natural systems, there are only rarely occasions when such a high number of sediment transport regimes may be preserved in and can be resolved from sedimentary deposits. The parameter l should be a vector between the minimum possible (zero) and maximum possible value (by definition the median, 0.5, but usually a lower value). Whensubmitting only a scalar, the variability can be only due to the range of possible endmembers (between q.min and q.max). If the parameter correct.output is enabled, this can decrease the number of valid values for l, i.e. the number of rows of the output matrix may no longer be the same as the length of the input vector of l. In such a case the vector l must be replaced by the rownames of the output matrix (l <- as.numeric(rownames(get.q()))).

Value

Numeric matrix of class "EMMAgeo_q", minimum and maximum numbers of end-members as well as corresponding weight transformation values as rownames.

Author(s)

Michael Dietze, Elisabeth Dietze

References


See Also

EMMA, test.parameters, test.robustness

Examples

## load example data set
data("example_X")

## create parameter matrix
get.q(X = X, l = c(0, 0.05, 0.10, 0.15))

Description

This function starts a browser-based graphic user interface for EMMA. The GUI has so far been tested on a Linux system, both with the browser of RStudio and Mozilla Firefox. It permits basic access to import, display and model a user-provided data set.

Usage

GUI(...)

Arguments

... further arguments to pass to runApp

Details

To use own data set, this should be a plain ASCII file with samples organised as rows and grain-size classes organised as columns. The ASCII file can be separated by spaces, commas, semi colons or tab stops. The file may contain a leading column with sample IDs and/or one leading row with grain-size class breaks. To run EMMA make sure that there are no classes that contain only zeros throughout all samples (i.e., remove them beforehand, e.g., by \( X = X[, \text{colSums}(X) > 0 \])).

Author(s)

Michael Dietze

See Also

runApp

Examples

```r
# Not run:
# Start the GUI
GUI()

# End(Not run)
```
interpolate.classes

Interpolate data between different classes.

Description

This function interpolates grain-size data for different classes, either to higher or to lower resolution.

Usage

interpolate.classes(
  X,
  boundaries.in,
  boundaries.out,
  method = "natural",
  fixed.start = TRUE
)

Arguments

- **X**: Numeric matrix, input data set with m samples (rows) and n variables (columns).
- **boundaries.in**: Numeric vector, class boundaries of the input data.
- **boundaries.out**: Numeric vector, class boundaries of the output data.
- **method**: Logical scalar, interpolation method, one out of "linear" (linear interpolation), "fmm" (cubic spline), "natural" (natural spline), "periodic" (periodic spline). Default is "natural".
- **fixed.start**: Logical scalar, specifying if the outer boundaries should be set to the same values as in the original matrix, default is TRUE. This may become necessary to avoid interpolation errors, see example.

Value

Numeric matrix, interpolated class values.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

EMMA, approx, spline
Examples

```r
## load example data
data(example_X)
classes.in <- seq(from = 1, to = 10, length.out = ncol(X))

## Example 1 - decrease the class numbers
## define number of output classes
classes.out <- seq(1, 10, length.out = 20)

## interpolate the data set
Y <- interpolate.classes(X = X,
                        boundaries.in = classes.in,
                        boundaries.out = classes.out,
                        method = "linear")

## show original vs. interpolation for first 10 samples
plot(NA, xlim = c(1, 10), ylim = c(0, 40))
for(i in 1:10) {
  lines(classes.in, X[i,] * 20 + i)
  lines(classes.out, Y[i,] * 20 + i, col = 2)
}

## Example 2 - increase the class numbers
## define number of output classes
classes.out <- seq(1, 10, length.out = 200)

## interpolate the data set
Y <- interpolate.classes(X = X,
                        boundaries.in = classes.in,
                        boundaries.out = classes.out)

## show original vs. interpolation for first 10 samples
plot(NA, xlim = c(1, 10), ylim = c(0, 40))
for(i in 1:10) {
  lines(classes.in, X[i,] * 20 + i)
  lines(classes.out, Y[i,] * 20 + i, col = 2)
}
```

mix.EM

Function to mix sample spectres.

Description

This functions allows to mix grain-size distributions with specified proportions and defined noise levels, for example to test the goodness of the EMMA algorithm.

Usage

mix.EM(EM, proportion, noise, autocorrelation)
Arguments

EM Numeric matrix, grain-size distribution definitions.
Each definition is in a separate row with variable contributions in columns.

proportion Numeric vector, relative proportions of each distribution per sample.

noise Numeric scalar, optional relative white noise levels.

autocorrelation Numeric scalar, the degree of autocorrelation among classes. Autocorrelation is realised as running mean of the specified length. Only odd values are allowed.

Details

The function multiplies each end-member with the respective proportion value, sums the resulting variables, adds uniform noise and normalises the resulting mixed sample to 100 %.

Value

Numeric vector, a sample composed of known proportions of end-members.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

create.EM

Examples

```r
## define end-member loadings and phi vector
EMa.1 <- create.EM(p1 = c(2, 8), p2 = c(1, 0.8), s = c(0.7, 0.3),
                   boundaries = c(0, 11), n = 80)
EMa.2 <- create.EM(p1 = c(4, 7), p2 = c(1.1, 1.4), s = c(0.5, 0.5),
                   boundaries = c(0, 11), n = 80)
EMa <- rbind(EMa.1, EMa.2)
phi <- seq(0, 11, length.out = 80)

## mix end-member loadings
sample1 <- mix.EM(EMa, proportion = c(0.3, 0.7))
sample2 <- mix.EM(EMa, proportion = c(0.5, 0.5), noise = 0.1,
                 autocorrelation = 3)

## plot end-member loadings (grey) and resulting samples (black)
plot(phi, EMa.1, type="l", col = "grey")
lines(phi, EMa.2, col = "grey")
lines(phi, sample1)
lines(phi, sample2)
```
model.EM

Model all possible end-member scenarios

Description

This function takes a definition of weight transformation limits and corresponding minimum and maximum numbers of end-members to model all end-member scenarios in accordance with these parameters. Based on the output the user can decide on robust end-members.

Usage

model.EM(X, q, l, classunits, plot = TRUE, col.q = TRUE, bw, ...)

Arguments

X      Numeric matrix, input data set with m samples (rows) and n variables (columns).
q      Numeric matrix, definitions of minimum and maximum number of end-members (cf. get.q()), required.
l      Numeric vector, weight transformation limit values, corresponding to the matrix q, required.
classunits Numeric vector, optional class units (e.g. micrometers or phi-units) of the same length as columns of X.
plot  Logical scalar, option to plot the results (cf. details for explanations), default is TRUE.
col.q  Logical scalar, option to colour end-member loadings by the number of end-members which were used to create the model realisation, default is TRUE.
bw    Numeric scalar, optional manual setting of the kde bandwidth. By default, bw is calculated as 1 percent of the number of grain-size classes.
...    Further arguments passed to the function.

Details

The plot output is an overlay of several data. The coloured lines in the background are end-member loadings (number noted in the plot title), resulting from all possible model scenarios. If col.q == TRUE they are coloured according to the number of end-members with which the model was generated. This colour scheme allows to depict end-members that emerge for model realisations with specific number of end-members. The thick black line is a kernel density estimate curve, generated from the mode positions of all end-members. The kernel bandwidth is set to 1 percent of the number of grain-size classes of the input data set, which gave useful results for most of our test data sets. The cumulative dot-line-plot is a further visualisation of end-member mode positions. The function is a modified wrapper function for the function test.robustness().

Value

List object with all modelled end-members, each described by input parameters, mode position, quality measures and value distributions.
residual.EM

Author(s)
Michael Dietze, Elisabeth Dietze

References

See Also
EMMA, test.1.max

Examples

```r
## load example data set
data(example_X)

## define input parameters
l <- c(0, 0.05, 0.10)
q <- cbind(c(2, 2, 3), c(5, 6, 4))

## infer l-vector
em_pot <- model.EM(X = X, q = q, l = l)
```

residual.EM(Vqn)

Calculate a residual end-member loading.

Description
This function calculates an optional residual end-member loading. It uses the modelled end-member loadings as input and evaluates the root of 1 minus the sum of all squared loadings. The residual end-member can be used to analyse the remaining variance, e.g. if not all (robust) EMs are included (cf. Dietze et al., 2012). Negative values are set to zero.

Usage
residual.EM(Vqn)

Arguments

Vqn Numeric matrix, m unscaled robust end-member loadings.

Value
Numeric vector, residual end-member loading.
Author(s)

Michael Dietze, Elisabeth Dietze

References


See Also

EMMA, robust.EM

Examples

```r
## load example data
data(example_X)
data(example_EMrob)

## define mean robust end-member loadings
Vqn <- EMMA(X = X, q = 2, plot = TRUE)$loadings

## perform residual end-member loading calculation
Vqn.res <- residual.EM(Vqn)

## model EMMA with the residual end-member
E_res <- EMMA(X = X,
   q = 3,
   Vqn = rbind(Vqn, Vqn.res),
   plot = TRUE)
```

Description

This function takes a list object with potential end-member loadings and extracts those with modes in specified limits to describe them by mean and standard deviation and use these descriptions to propagate the uncertainties to end-member scores.

Usage

```r
robust.EM(
   em,
   limits,
   classunits,
)```
`robust.EM`  

```r
amount,
l,
mc_n,
type = "mean",
qt = c(0.25, 0.75),
cores = 1,
plot = FALSE,
...)
```

**Arguments**

- `em` List of class "EMMAgeo_empot", i.e. the output of `model.em()` or `test.robustness()`, containing potential end-members, both in unscaled and rescaled version as well as further parameters.
- `limits` Numeric matrix with two columns, defining the class limits for the robust end-members to calculate. The first column defines the lower limits, the second column the upper limits. End-members are organised in rows.
- `classunits` Numeric vector, optional class units (e.g. micrometers or phi-units) of the same length as columns of `X`.
- `amount` Numeric matrix with two columns, defining the minimum and maximum amount of the modal class for each end-member.
- `l` Numeric scalar, weight transformation limit for modelling the average end-member output.
- `mc_n` Numeric scalar, number of Monte Carlo simulations to estimate end-member scores uncertainty. The default setting is ten times the product of number of end-members and number of weight transformation limits. The latter is inherited from `model.em()`. To disable modelling of scores uncertainty, set `mc_n = 0`.
- `type` Character scalar, type of loadings statistics. One out of "mean" and "median". Default is "mean".
- `qt` Numeric vector of length two, quantiles to describe end-member loadings. Default is `c(0.25, 0.75)` (i.e., the quartile range).
- `cores` Numeric scalar, number of CPU cores to be used for calculations. Only useful in multicore architectures. Default is 1 (single core).
- `plot` Logical scalar, option for plot output. Default is FALSE.
- `...` Additional arguments passed to `EMMA` and `plot`.

**Details**

The function is used to extract potential end-member loadings based on their mode positions and, optionally the height of the mode class, and use them to infer mean and standard deviation of all end-members that match the group criteria defined by `limits`. These information are then used to model the uncertainty of the corresponding end-member scores. The function uses input from two preceeding approaches. In a compact protocol `model.em` delivers these data in a predefined way. In the extended protocol `test.robustness` does this.
Value

List with statistic descriptions of end-member loadings and scores.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

robust.loadings, robust.scores

Examples

## Not run:
## load example data set
data(example_X)
## get weight transformation limit vector
l <- get.l(X = X)
## get minimum and maximum number of end-members
q <- get.q(X = X, l = l)
## get all potential model scenarios
EM_pot <- model.EM(X = X, q = q, plot = TRUE)
## define end-member mode class limits
limits <- cbind(c(61, 74, 95, 102),
                c(64, 76, 100, 105))
## get robust end-members in the default way, with plot output
rem <- robust.EM(em = EM_pot,
               limits = limits,
               plot = TRUE)
## get robust end-members by only modelling uncertainty in loadings
robust_EM <- robust.EM(em = EM_pot,
                limits = limits,
                plot = TRUE)

## End(Not run)
robust.loadings

Extract robust end-member loadings

Description

This function takes a list object with potential end-member loadings and extracts those with modes in specified limits to describe them by mean and standard deviation.

Usage

```r
robust.loadings(
  em,  
  limits, 
  classunits,  
  amount,  
  type = "mean", 
  qt = c(0.25, 0.75), 
  plot = FALSE, 
  ...
)
```

Arguments

- `em` List of class "EMMAgeo_empot", i.e. the output of `model.EM()` or `test.robustness()`, containing potential end-members, both in unscaled and rescaled version as well as further parameters.
- `limits` Numeric matrix with two columns, defining the class limits for the robust end-members to calculate. The first column defines the lower limits, the second column the upper limits. End-members are organised in rows.
- `classunits` Numeric vector, optional class units (e.g. micrometers or phi-units) of the same length as the number of (grain-size) classes per sample.
- `amount` Numeric matrix with two columns, defining the minimum and maximum amount of the modal class for each end-member.
- `type` Character scalar, type of statistics. One out of "mean" and "median". Default is "mean".
- `qt` Numeric vector of length two, quantiles to describe end-member loadings. Default is c(0.25, 0.75) (i.e., the quartile range).
- `plot` Logical scalar, option to enable plot output. Default is FALSE.
- `...` Additional arguments passed to `EMMA` and `plot`.

Value

List with statistic descriptions of unscaled and scaled end-member loadings.
robust.scores

Author(s)
Michael Dietze, Elisabeth Dietze

See Also
robust.EM, robust.scores

Examples

```r
## load example data set, potential end-members, output of model.EM()
data(example_EMpot)

## define limits for robust end-members
limits <- cbind(c(61, 74, 95, 102),
                c(64, 76, 100, 105))

## get robust end-member loadings with plot output
robust_loadings <- robust.loadings(em = EMpot,
                                    limits = limits,
                                    plot = TRUE)
```

Description
This function takes a list object with statistics of end-member loadings and propagates these uncertainties to end-member scores using Monte Carlo methods.

Usage

```r
robust.scores(loadings, l, mc_n, cores = 1, plot = FALSE, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loadings</td>
<td>List of class “EMMAgeo_roblload”, i.e. the output of robust.loadings(), containing statistic descriptions of robust end-member loadings.</td>
</tr>
<tr>
<td>l</td>
<td>Numeric scalar, weight transformation limit to use for modelling the average end-member output. Can be output of get.l.opt(). If omitted, it is set to 0.</td>
</tr>
<tr>
<td>mc_n</td>
<td>Numeric scalar, number of Monte Carlo simulations to estimate end-member scores uncertainty. The default setting is ten times the product of number of end-members and number of weight transformation limits. The latter is inherited from model.em(). To disable modelling of scores uncertainty, set mc_n = 0.</td>
</tr>
<tr>
<td>cores</td>
<td>Numeric scalar, number of CPU cores to be used for calculations. Only useful in multicore architectures. Default is 1 (single core).</td>
</tr>
<tr>
<td>plot</td>
<td>Logical scalar, option for plot output. Default is FALSE.</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments passed to EMMA and plot.</td>
</tr>
</tbody>
</table>
Value

List with statistic descriptions of robust end-member scores.

Author(s)

Michael Dietze, Elisabeth Dietze

See Also

robust.EM, robust.loadings

Examples

```r
## load example data set, potential end-members, output of model.EM()
data(example_EMpot)

## define limits for robust end-members
limits <- cbind(c(61, 74, 95, 102),
                c(64, 76, 100, 105))

## get robust end-member loadings
robust_loadings <- robust.loadings(em = EMpot, limits = limits)

## model end-member scores uncertainties with minimum Monte Carlo runs
robust_scores <- robust.scores(loadings = robust_loadings,
                                mc_n = 5,
                                plot = TRUE)
```

test.factors Calculate the initial cumulative explained variance of factors.

Description

This function performs eigenspace decomposition using the weight-transformed matrix W to determine the explained variance with increasing number of factors. Depending on the number of provided weight transformation limits (l) a vector or a matrix is returned.

Usage

test.factors(X, l, c, r.min = 0.95, plot = FALSE, legend, ...)
Arguments
X Numeric matrix, input data set with m samples (rows) and n variables (columns).
l Numeric vector, weight transformation limits, i.e. quantiles; default is 0.
c Numeric scalar, constant sum scaling parameter, e.g. 1, 100, 1000; default is 100.
r.min Numeric scalar, minimum value of explained variance to be reached by the end-members included, default is 0.95.
plot Logical scalar, optional graphical output of the results, default is FALSE.
legend Character scalar, specify legend position (cf. legend). If omitted, no legend will be plotted, default is no legend.
... Additional arguments passed to the plot function. Use colour instead of col to create different colours.

Details
The results may be used to define a minimum number of end-members for subsequent modelling steps, e.g. by using the Kaiser criterion, which demands a minimum number of eigenvalues to reach a squared R of 0.95.

Value
List with objects
L Vector or matrix of cumulative explained variance.
q.min Vector with number of factors that passed r.min.

Author(s)
Michael Dietze, Elisabeth Dietze

References

Examples
## load example data set
data(example_X)

## create sequence of weight transformation limits
l <- seq(from = 0, to = 0.2, 0.02)

## perform the test and show q.min
L <- test.factors(X = X, l = l, c = 100, plot = TRUE)
## a visualisation with more plot parameters
L <- test.factors(X = X, l = 1, c = 100, plot = TRUE,
                   ylim = c(0.5, 1), xlim = c(1, 7),
                   legend = "bottomright", cex = 0.7)

## another visualisation, a close-up
plot(1:7, L$L[1,1:7], type = "l",
     xlab = "q", ylab = "Explained variance")
for(i in 2:7) {lines(1:7, L$L[i,1:7], col = i)}

---

### Description

This function performs the weight transformation of the data matrix after Klovan & Imbrie (1971) and performs EMMA() with different weight limits to check if valid results are yielded. It returns the maximum value for which the transformation remains stable.

### Usage

test.l(X, l, ...)

### Arguments

- **X**
  - Numeric matrix, input data set with m samples (rows) and n variables (columns).
- **l**
  - Numeric vector, weight transformation limit, i.e. quantile; default is 0.
- **...**
  - Further arguments passed to the function.

### Value

- **List** with objects
  - **step**
    - Numeric scalar with position of the last valid value.
  - **l.max**
    - Numeric scalar with last valid value of l.

### Author(s)

Michael Dietze, Elisabeth Dietze
References


See Also

EMMA, check.data, test.parameters

Examples

```r
## load example data set
data(example_X)

test <- test.l(X = X, l = seq(from = 0, to = 0.6, by = 0.1))
```

---

test.l.max  

*Find maximum possible wight transformation value.*

Description

This function approximates the highest possible value for l in a nested loop. It uses test.l and does not need any further parameters. It starts with l between zero and 0.5 and iteratively approximates the last possible values for which the weight-transformed matrix of the input data still allows eigenspace extraction.

Usage

```r
test.l.max(X, n = 10, ...)
```

Arguments

- **X**  
  Numeric matrix, input data set with m samples (rows) and n variables (columns).

- **n**  
  Numeric scalar, number of loop runs and values per loop.

- **...**  
  Further arguments passed to the function.

Value

Numeric scalar, maximal possible l value.
**test.parameters**

Evaluate influence of model parameters.

**Description**

All possible combinations of number of end-members and weight transformation limits are used to perform EMMA and evaluate the absolute and relative measures of individual model performance.

**Usage**

```r
test.parameters(
  X,
  q,
  l = 0,
  c = 100,
  rotation = "Varimax",
  plot = FALSE,
)```

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**


**See Also**

EMMA, test.l

**Examples**

```r
## load example data set
data(example_X)

## create weight transformation limits vector
l <- seq(from = 0, to = 0.6, by = 0.02)

## test l.max
l.max <- test.l.max(X = X)
```
Legends, multicore = FALSE, ...

Arguments

- **X**  
  Numeric matrix, input data set with m samples (rows) and n variables (columns).

- **q**  
  Numeric vector, numbers of end-members to be modelled, e.g., 2:10.

- **l**  
  Numeric vector specifying the weight transformation limit, i.e. quantile; default is 0.

- **c**  
  Numeric scalar specifying the constant sum scaling parameter, e.g., 1, 100, 1000; default is 0.

- **rotation**  
  Character scalar, rotation type, default is "Varimax".

- **plot**  
  Character scalar, optional graphical output of the results as keyword (see details). All plots except "ol" are colour-coded bitmaps of q, l and the specified test parameter and line-plots the specified parameter vs. q.

- **legend**  
  Character scalar, specifying legend position (cf. legend). If omitted, no legend will be plotted, default is no legend.

- **multicore**  
  Logical scalar, optionally distribute calculations to all available cores of the computer, default is TRUE.

- **...**  
  Additional arguments passed to the plot function (see details).

Details

The mean total explained variance \( mR_t \) may be used to define a maximum number of meaningful end-members for subsequent modelling, e.g. as the number of end-members, which reaches the first local \( mR_t \) maximum.

Overlapping is defined as one end-member having its mode within the "area" of any other end-member, which is genetically not explainable.

Keywords to specify, which tested parameter will be plotted: "mEm" (mean absolute row-wise error), "mEn" (mean absolute column-wise error), "mRm" (mean relative row-wise error), "mRn" (mean relative column-wise error), "mRt" (mean relative total error) and "ol" (number of overlapping end-members).

Since the function returns two plots (except for option "ol"), additional graphical parameters must be specified as vector with the first element for the first plot and the second element for the second plot. If graphical parameters are natively vectors (e.g. a sequence of colours), they must be specified as matrices with each vector as a row. A legend can only be added to the second plot. Colours only apply to the second plot as well. If colours are specified, colour should be used instead of col. See example section for further advice.

Value

- **List** with result objects

  - **mEm**  
    Absolute row-wise model error.

  - **mEn**  
    Absolute column-wise model error.
test.parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mRm</td>
<td>Mean row-wise explained variance.</td>
</tr>
<tr>
<td>mRn</td>
<td>Mean column-wise explained variance.</td>
</tr>
<tr>
<td>mRt</td>
<td>Mean total explained variance.</td>
</tr>
<tr>
<td>o1</td>
<td>Number of overlapping end-member loadings.</td>
</tr>
<tr>
<td>q.max</td>
<td>Maximum number of meaningful end-members.</td>
</tr>
</tbody>
</table>

Author(s)

Michael Dietze, Elisabeth Dietze

References


See Also

EMMA

Examples

```r
## load example data set
data(example_X)

## truncate the data set for faster computation
X.trunc <- X[1:20,]

## define test parameters
q <- 2:8 # number of end-members
l <- seq(from = 0, to = 0.3, by = 0.1)

## test parameter influence and plot mean total explained variance
TP <- test.parameters(X = X.trunc, q = q, l = l, plot = "mRt",
                      legend = "bottomright", cex = 0.7,
                      multicore = FALSE,
                      colour = rgb((1:7) / 7, 0.9, 0.2, 1))

## show maximum number of end-members
TP$q.max
```
test.robustness (Test model robustness.)

Description

This function takes a definition of weight transformation limits and corresponding minimum and maximum numbers of end-members to model all end-member scenarios in accordance with these parameters. Based on the output the user can decide on robust end-members.

Usage

test.robustness(
  X,
  q,
  l,
  P,
  c,
  classunits,
  ID,
  rotation = "Varimax",
  ol.rej,
  mRt.rej,
  plot = FALSE,
  ...
)

Arguments

X  Numeric matrix with m samples (rows) and n variables (columns).
q  Numeric vector with number of end-members to be modelled.
l  Numeric vector specifying the weight tranformation limits, i.e. quantiles; default is 0.
P  Numeric matrix, optional alternative input parameters for q and l, either of the form m:3 with m variations in the columns q.min, q.max, l or of the form m:2 with m variations in the columns q, l.
c  Numeric scalar specifying the constant sum scaling parameter, e.g. 1, 100, 1000; default is 100.
classunits  Numeric vector, optional class units (e.g. phi classes or micrometers) of the same length as columns of X.
ID  Numeric or character vector, optional sample IDs of the same length as columns of X.
rotation  Character scalar, rotation type, default is "Varimax" (cf. Dietze et al., 2012). One out of the rotations provided in GPArotation is possible (cf. rotations).
ol.rej Numeric scalar, optional rejection threshold for overlapping criterion. All model runs with overlapping end-members greater than the specified integer will be removed.

mRt.rej Numeric scalar, optional rejection threshold for mean total explained variance criterion. All modelled end-members below the specified value will be removed.

plot Logical scalar, optional graphical output of the results, default is FALSE. If set to TRUE, end-member loadings and end-member scores are plotted.

Details
The function value $\text{loadings}$ is redundant but was added for user convenience. Since the function returns two plots, additional graphical parameters must be specified as vector with the first element for the first plot and the second element for the second plot. If graphical parameters are natively vectors (e.g. a sequence of colours), they must be specified as matrices with each vector as a row. If colours are specified, colour should be used instead of col. ylim can only be modified for the first plot. See example section for further advice.

Value
A list with objects

- q Vector with q.
- l Vector with l.
- modes Vector with mode class.
- mRt Vector with mean total explained variance.
- ol Vector with n overlapping end-members.
- loadings Matrix with normalised rescaled end-member loadings.
- Vqsn Matrix with rescaled end-member loadings.
- Vqn Matrix with normalised factor loadings.

Author(s)
Michael Dietze, Elisabeth Dietze

References
## Examples

```r
## load example data set
data(example_X)

## Example 1 - perform the most simple test
q <- 4:7
l <- seq(from = 0, to = 0.1, by = 0.02)
M1 <- test.robustness(X = X, q = q, l = l,
                      ol.rej = 1, mRt.rej = 0.8,
                      plot = TRUE,
                      colour = c(4, 7),
                      xlab = c(expression(paste("Grain size (", phi, ", "))),
                              expression(paste("Grain size (", phi, ", ", sep = ""))))

## Example 2 - perform the test without rejection criteria and plots
P <- cbind(rep(q[1], length(l)),
           rep(q[3], length(l)),
           l)
M2 <- test.robustness(X = X, P = P)

## Plot 1 - end-member loadings which do not overlap and yielded mRt > 0.80.
plot(M2$Vqsn[1,], type = "l", ylim = c(0, max(M2$Vqsn, na.rm = TRUE)),
     main = "End-member loadings")
for (i in 2:nrow(M2$Vqsn)) lines(M2$Vqsn[i,])

## Plot 2 - histogram of mode positions
hist(M2$modes,
     breaks = 1:ncol(X),
     main = "Mode positions",
     xlab = "Class")

## Plot 3 - positions of modelled end-member modes by number of end-members
# Note how scatter in end-member position decreases for the "correct" number
# of modelled end-members (6) and an appropriate weight limit (ca. 0.1).
ii <- order(M2$q, M2$modes)

modes <- t(rbind(M2$Vqsn, M2$q))[ii,]
plot(modes[,1],
     seq(1, nrow(modes)),
     main = "Model overview",
     xlab = "Class",
     ylab = "EM number in model run",
     pch = as.character(modes[,2]),
     cex = 0.7)

# Illustrate mode positions as stem-and-leave-plot, useful as a simple
# check, which mode maxima are consistently fall into which grain-size
# class (useful to define "limits" in robust.EM).
stem(M2$modes, scale = 2)
```
**Description**

Synthetic data set created by randomly mixed natural end-members

**Format**

```
num [1:100, 1:116] 0.000899 0.000516 0.00136 0.000989 0.00102 ...
```

**Details**

The dataset is the result of four mixed natural end-members.

**Examples**

```r
## load example data set
data(example_X)

## extract grain-size classes
s <- as.numeric(colnames(X))

## plot first 10 samples stacked in one line plot
plot(NA,
    xlim = c(1, ncol(X)),
    ylim = c(1, 20))

for(i in 1:10) {
    lines(x = s,
          y = X[i,] + i)
}

## plot grain-size map
image(x = s,
      z = t(X),
      log = "x",
      col = rainbow(n = 250))
```
Index

* EMMA
  check.data, 3
  click.limits, 4
  convert.units, 5
  create.EM, 6
  EMMA, 7
  get.l, 11
  get.l.opt, 12
  get.limits, 13
  get.q, 14
  interpolate.classes, 17
  mix.EM, 18
  model.EM, 20
  residual.EM, 21
  robust.EM, 22
  robust.loadings, 25
  robust.scores, 26
  test.factors, 27
  test.l, 29
  test.l.max, 30
  test.parameters, 31
  test.robustness, 34

* datasets
  EMpot, 10
  EMrob, 10
  X, 37

* package
  EMMAgeo-package, 2
  approx, 17
  check.data, 3, 30
  click.limits, 4
  convert.units, 5
  create.EM, 6, 19
  eigen, 9
  EMMA, 3, 7, 12, 14, 15, 17, 21, 22, 30, 31, 33
  EMMAgeo (EMMAgeo-package), 2
  EMMAgeo-package, 2
  EMpot, 10
  EMrob, 10
  get.l, 11
  get.l.opt, 12
  get.limits, 13
  get.q, 14
  GUI, 16
  interpolate.classes, 5, 17
  legend, 28, 32
  mix.EM, 7, 18
  model.EM, 14, 20
  nnls, 9
  residual.EM, 21
  robust.EM, 4, 22, 22, 26, 27
  robust.loadings, 24, 25, 27
  robust.scores, 24, 26, 26
  rotations, 9, 34
  runApp, 16
  spline, 17
  test.factors, 27
  test.l, 29, 31
  test.l.max, 11, 21, 30
  test.parameters, 9, 15, 30, 31
  test.robustness, 4, 15, 34
  X, 37