

# Package ‘EMMAgeo’

July 2, 2014

**Type** Package

**Title** End-member modelling algorithm and supporting functions for grain-size analysis

**Version** 0.9.1

**Date** 2013-10-14

**Author** Michael Dietze, Elisabeth Dietze

**Maintainer** Michael Dietze <micha.dietze@mailbox.tu-dresden.de>

**Description** This package provides a set of functions for convenient end-member modelling analysis of grain-size data (EMMAgeo).

**License** GPL-3

**Depends** R (>= 3.0.1), GPArotation, limSolve, shape

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2013-10-14 14:16:18

## R topics documented:

EMMAgeo-package . . . . .	2
check.data . . . . .	2
convert.units . . . . .	4
create.EM . . . . .	5
define.limits . . . . .	6
EMMA . . . . .	7
interpolate.classes . . . . .	9
mix.EM . . . . .	11
Mqs.uncertainty . . . . .	12
residual.EM . . . . .	15
robust.EM . . . . .	16
test.L . . . . .	18

test.lw . . . . .	20
test.parameters . . . . .	21
test.robustness . . . . .	23
X.artificial . . . . .	26

<b>Index</b>	<b>27</b>
--------------	-----------

---

EMMAgeo-package	<i>End-member modelling algorithm and supporting functions for grain-size analysis</i>
-----------------	--

---

## Description

This package provides a set of functions for convenient end-member modelling analysis of grain-size data (EMMAgeo).

## Details

Package: EMMAgeo  
 Type: Package  
 Title: End-member modelling algorithm and supporting functions for grain-size analysis  
 Version: 0.9.1  
 Date: 2013-10-14  
 Author: Michael Dietze, Elisabeth Dietze  
 Maintainer: Michael Dietze <micha.dietze@mailbox.tu-dresden.de>  
 License: GPL-3  
 Depends: R (>= 3.0.1), GPArotation, limSolve, shape

## Author(s)

Michael Dietze, Elisabeth Dietze

---

check.data	<i>Function to check data consistency.</i>
------------	--

---

## Description

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

**Usage**

```
check.data(X, q, lw, c, invisible = TRUE)
```

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
q	Numeric scalar with number of end-members to be modelled.
lw	Numeric scalar or vector specifying the weight transformation limit, i.e. quantile.
c	Numeric scalar specifying the constant sum scaling parameter, e.g. 1, 100, 1000.
invisible	Logical scalar setting visibility option.

**Value**

Character vector with test results.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**

Dietze E, Hartmann K, Diekmann B, Ijmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

**See Also**

[EMMA](#)

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

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

---

convert.units	<i>Function to convert between phi and micrometers.</i>
---------------	---

---

### Description

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

### Usage

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

### Arguments

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

### Value

Numeric vector with converted grain-size class values.

### Author(s)

Michael Dietze, Elisabeth Dietze

### See Also

[interpolate.classes](#)

### Examples

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

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

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

---

create.EM                      *Function to create grain-size-distributions.*

---

### Description

This function allows to create artificial grain-size-compositions. 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 with means of normal distributions, i.e. mode positions.
p2	Numeric vector with standard deviations of normal distributions, i.e. mode width.
s	Numeric vector with relative proportions of each mode, i.e. relative mode height.
boundaries	Numeric vector of length 2 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 mixing individual artificial end member loadings, these should span over the same classes. Hence, `boundaries` and `n` should be the same for all end-member loadings. 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

### See Also

[mix.EM](#)

**Examples**

```
## 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")
```

---

define.limits

*Function to define limits by mouse clicks.*


---

**Description**

This function allows to define 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**

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

**Arguments**

data	Output of <code>test.robustness</code> , a list with several objects.
n	Numeric scalar with number of target end-members (i.e. half the number of limits).
classunits	Numeric vector, optional class units (e.g. micrometers or phi-units).

**Value**

Numeric matrix with 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](#)

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

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

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

---

EMMA

---

*Function for end-member modelling analysis.*


---

**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, eigen space decomposition, factor rotation, data modelling and evaluation are performed.

**Usage**

```
EMMA(X, q, lw, c, Vqn, EM.ID, classunits, ID, rotation = "Varimax",
      plot = FALSE, legend, ..., pm = FALSE)
```

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
q	Numeric scalar with number of end-members to be modelled.
lw	Numeric scalar with the weight transformation limit, i.e. quantiles, cf. Klován & Imbrie (1971); default is 0.
c	Numeric scalar specifying the constant sum scaling parameter, e.g. 1, 100, 1000; default is 100.
Vqn	Numeric matrix specifying optional unscaled user-defined end-member loadings. If provided, these are used instead of model-derived ones.
EM.ID	Character vector with end-member names. If present, these will be set as row-names of the output data set and used in the legend text if a legend is enabled.
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.
rotation	Character scalar, rotation type, default is "Varimax" (cf. Dietze et al., 2012). One out of the rotations provided in GPArotation is possible (cf. <a href="#">rotations</a> ).
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.
legend	Character scalar, legend position (cf. <a href="#">legend</a> ). If omitted, no legend will be plotted, default is no legend.
...	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. Legend contents and colours apply to both plots. If colours are specified, colour should be used instead of col. See example section for further advice.
pm	Logical scalar to enable pm.

### Details

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

### Author(s)

Michael Dietze, Elisabeth Dietze



## References

- Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.
- Klovan JE, Imbrie J. 1971. An Algorithm and FORTRAN-IV Program for Large-Scale Q-Mode Factor Analysis and Calculation of Factor Scores. *Mathematical Geology* 3: 61-77.
- Miesch AT. 1976. Q-Mode factor analysis of geochemical and petrologic data matrices with constant row sums. U.S. Geological Survey Professional Papers 574.

## See Also

[test.parameters](#), [rotations](#), [eigen](#), [npls](#)

## Examples

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

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

## perform EMMA with 4 end-members and more graphical settings
EM <- EMMA(X = X.artificial, q = 4, lw = 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"),
           legend = "topleft",
           cex = 0.7,
           colour = colors()[c(441, 496, 499, 506)])
```

---

`interpolate.classes`     *Function to interpolate classes.*

---

## Description

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

## Usage

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

**Arguments**

<code>X</code>	Numeric matrix with <code>m</code> samples (rows) and <code>n</code> variables (columns).
<code>boundaries.in</code>	Numeric vector with class boundaries of the input data.
<code>boundaries.out</code>	Numeric vector with class boundaries of the output data.
<code>method</code>	Logical scalar, interpolation method, one out of "linear" (linear interpolation), "fmm" (cubic spline), "natural" (natural spline), "periodic" (periodic spline). Default is "natural".
<code>fixed.start</code>	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 with interpolated class values.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**

Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

**See Also**

[EMMA](#), [approx](#), [spline](#)

**Examples**

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

## 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.artificial,
                        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, 11))
for(i in 1:10) {
```

```

    lines(classes.in, X.artificial[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.artificial,
                          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, 11))
  for(i in 1:10) {
    lines(classes.in, X.artificial[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 containing the grain-size distribution definitions. Each definition is in a separate row with variable contributions in columns.
proportion	Numeric vector containing the relative proportions of each distribution per sample.
noise	Numeric scalar containing optional relative white noise levels.
autocorrelation	Numeric scalar specifying 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 comprising a sample composed of known proportions of end-members.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**See Also**

[create.EM](#)

**Examples**

```
## 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)
```

---

Mqs.uncertainty

*Function to estimate end-member scores uncertainty*

---

**Description**

The function uses either existing assemblages of end-member loadings or specified measures of centrality and dispersion as input for Monte Carlo runs to estimate the influence of different end-member loadings on end-member scores. Likewise, the influence of the weight limit quantiles (lw) can be estimated.

**Usage**

```
Mqs.uncertainty(X, q, lw, c, rotation = "Varimax", Vqn, Vqn.sd,
  runs, type.lw, autocorrelation)
```

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
q	Numeric scalar with the number of end-members to include. Only necessary in combination with Vqn as matrix of user-defined end-member loadings.
lw	Numeric vector with the weight transformation limits (i.e. quantiles after Klován & Imbrie, 1971). If the parameter is of length 1, lw is assumed to be a constant, if of length 2, lw defines either mean and standard deviation or minimum and maximum, depending on the value of type.lw.
c	Numeric scalar specifying the constant sum scaling parameter, e.g. 1, 100, 1000; default is 100.
rotation	Character scalar, rotation type, default is "Varimax" (cf. Dietze et al., 2012). One out of the rotations provided in GPArotation is possible (cf. <a href="#">rotations</a> ).
Vqn	Numeric matrix with existing unscaled end-member loadings. These may represent user-defined loadings (or mean loadings if Vqn.sd is specified). See example section for further information.
Vqn.sd	Numeric matrix with standard deviations of the mean unscaled end-member loadings in Vqn.
runs	Logical scalar with the number of Monte Carlo runs to be performed, default is 100.
type.lw	Character scalar with type of random lw value generation. Either "rnorm" or "runif", default is "runif".
autocorrelation	Numeric scalar optionally specifying the degree of autocorrelation among classes. Autocorrelation is realised as running mean of the specified length. Only odd values are allowed.

**Value**

A list with numeric vector and matrix objects.

lw	Randomised weight limit values.
Vqn	Randomised unscaled end-member loadings.
Mqs	Modelled end-member scores.
mean	Modelled end-member score means.
sd	Modelled end-member score standard deviations.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**

Dietze E, Hartmann K, Diekmann B, Ijmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake

sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

Klovan JE, Imbrie J. 1971. An Algorithm and FORTRAN-IV Program for Large-Scale Q-Mode Factor Analysis and Calculation of Factor Scores. *Mathematical Geology* 3: 61-77.

### See Also

[test.robustness](#), [test.parameters](#)

### Examples

```
## load example data set
data(X.artificial, envir = environment())

## set model run parameters
q = 3 # set number of end-members, try 4 to see the difference!
Vqn <- EMMA(X.artificial, q)$Vqn # assign unscaled end-member loadings
Vqn.sd <- Vqn * 0.2 # assign a relative standard deviation of 20 %
lw.1 <- 0.2 # set lw to 0.2
lw.2 <- c(0.2, 0.08) # set lw to mean = 0.2 and sd = 0.08
runs <- 12 # senseless value to increase computation speed

## EXAMPLE 1
## Calculate Mqs uncertainty
M <- Mqs.uncertainty(X = X.artificial,
                    q = q,
                    lw = lw.1,
                    runs = runs,
                    Vqn = Vqn,
                    Vqn.sd = Vqn.sd,
                    type.lw = "rnorm",
                    autocorrelation = 3)

## Plot line-point graph with means and standard deviations
plot(NA,
     xlim = c(1, nrow(X.artificial)),
     ylim = c(0.5, q + 1),
     main = "End-member scores with uncertainty")
for(i in 1:q) {
  lines(1:nrow(X.artificial), M$mean[,i] - M$sd[,i] + i, col = i, lty = 2)
  lines(1:nrow(X.artificial), M$mean[,i] + i, col = i, lwd = 2)
  points(1:nrow(X.artificial), M$mean[,i] + i, col = i)
  lines(1:nrow(X.artificial), M$mean[,i] + M$sd[,i] + i, col = i, lty = 2)
}

## EXAMPLE 2
## Calculate Mqs uncertainty
M <- Mqs.uncertainty(X = X.artificial,
                    q = q,
                    lw = lw.2,
                    runs = runs,
                    Vqn = Vqn,
```

```
        type.lw = "rnorm")

## Plot point graph with error bars
plot(NA,
     xlim = c(1, nrow(X.artificial)),
     ylim = c(0.5, q + 1),
     main = "End-member scores with uncertainty")
for(i in 1:q) {
  points(1:nrow(X.artificial), M$mean[,i] + i, pch = 3, col = i)
  arrows(1:nrow(X.artificial), M$mean[,i] - M$sd[,i] + i,
        1:nrow(X.artificial), M$mean[,i] + M$sd[,i] + i,
        code = 3, angle = 90, length = 0.05, col = i)
}
```

---

`residual.EM`*Function to evaluate 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 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 with m robust end-member loadings.

## Value

Numeric vector with residual end-member loading.

## Author(s)

Michael Dietze, Elisabeth Dietze

## References

Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

## See Also

[EMMA](#), [robust.EM](#)

## Examples

```
## Some preparing steps to retrieve only robust end-members
## load example data
data(X.artificial, envir = environment())

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

## define limits data set
limits = cbind(c(11, 31, 60, 78),
               c(13, 33, 62, 80))

## define end-member numbers to test
q <- 4:8
## define weight transformation limits
lw <- seq(from = 0, to = 0.1, by = 0.05)
## perform the robustness test
TR <- test.robustness(X.trunc, q, lw, c = 100)

## extract robust end-members
REM <- robust.EM(Vqsn = TR$Vqsn, limits = limits, Vqn = TR$Vqn)

## Calculation of residual end-member loading
## define mean robust end-member loadings
Vqn.rob <- REM$Vqn.mean
## perform residual end-member loading calculation
Vqn.res <- residual.EM(Vqn.rob)

# Visualisation of the result
plot(NA, xlim = c(1, 80), ylim = c(0, 1))
for(i in 1:4) {lines(Vqn.rob[i,])}
lines(Vqn.res, col = 2)
```

---

robust.EM

*Function to extract robust end-members.*

---

## Description

This function takes a matrix with end-member loadings and extracts those whose modes fall into specified limits. The function returns a list with all passing end-member loadings and scores, along with their respective column-wise (variable-wise) measures of centrality and dispersion.

## Usage

```
robust.EM(Vqsn, limits, quantiles, Vqn, classunits, ID, plot = FALSE,
          legend, ..., pm = FALSE)
```



**Arguments**

Vqsn	Numeric matrix with m samples (rows) and n variables (columns).
limits	Numeric matrix with two columns that contain the boundaries of mode classes for each end-member. The first column contains the lower, the second column the upper limit. If <code>classunits</code> are provided, the limits are assumed to relate to these units, if omitted column-numbers of <code>Vqsn</code> are used.
quantiles	Optional numeric vector of length two with the quantiles to be evaluated for the robust end-member loadings; default is <code>c(0.25, 0.75)</code> .
Vqn	Numeric matrix with optional normalised factor loadings. If present, the same factor loadings as the respectively selected end-member loadings are returned.
classunits	Numeric vector, optional class units (e.g. <code>phi</code> classes or micrometers) of the same length as columns of <code>X</code> .
ID	Numeric or character vector, optional sample IDs of the same length as columns of <code>X</code> .
plot	Logical scalar, optional graphical output of the results, default is <code>FALSE</code> . If set to <code>TRUE</code> , selected end-member loadings are plotted in different colours, according to the specified classes. All end-member loadings are plotted in pale colour, means and standard deviations are plotted above in thicker lines. To plot median and quantile range instead of mean and standard deviation, add <code>median = TRUE</code> as further plot parameter. See examples section for further advice.
legend	Character scalar, specifying legend position (cf. <a href="#">legend</a> ). If omitted, no legend will be plotted, default is no legend.
...	Additional arguments passed to the plot function. Use <code>colour</code> instead of <code>col</code> to create different colours.
pm	Logical scalar to enable pm.

**Value**

A list object containing:

Vqsn.data	A list with Vqsn values.
Vqsn.mean	A matrix with Vqsn means.
Vqsn.median	A matrix with Vqsn medians.
Vqsn.sd	A matrix with Vqsn standard deviations.
Vqsn.qt1	A matrix with Vqsn quantiles 1.
Vqsn.qt2	A matrix with Vqsn quantiles 2.
Vqn.data	A list with Vqn values.
Vqn.mean	A matrix with Vqn means.
Vqn.median	A matrix with Vqn medians.
Vqn.sd	A matrix with Vqn standard deviations.
Vqn.qt1	A matrix with Vqn quantiles 1.
Vqn.qt2	A matrix with Vqn quantiles 2.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**

Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

**See Also**

[EMMA](#), [test.robustness](#), [define.limits](#)

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

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

## define limits data set
limits = cbind(c(11, 31, 60, 78),
               c(13, 33, 62, 80))

## end-member numbers to test
q <- 4:8
## weight transformation limits to test
lw <- seq(from = 0, to = 0.1, by = 0.025)

## perform robustness test without rejection criteria and plots
TR <- test.robustness(X.trunc, q, lw)
## extract end-member loadings from robustness test data set
Vqsn <- TR$Vqsn

## extract robust end-members with limits matrix
REM <- robust.EM(Vqsn = Vqsn, limits = limits,
                plot = TRUE,
                legend = "topleft",
                cex = 0.7,
                colour = c("orange", "navyblue", "springgreen4", "red4"),
                median = TRUE)
```

## Description

This function performs eigenspace decomposition using the weight-transformed matrix  $W$  to determine the minimum number of end-members. Depending on the number of provided weight transformation limits ( $lw$ ) a single vector or a matrix is returned.

## Usage

```
test.L(X, lw, c, r.min = 0.95, plot = FALSE, legend, ..., pm = FALSE)
```

## Arguments

$X$	Numeric matrix with $m$ samples (rows) and $n$ variables (columns).
$lw$	Numeric vector specifying the weight transformation limits, i.e. quantiles; default is 0.
$c$	Numeric scalar specifying the 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. <a href="#">legend</a> ). If omitted, no legend will be plotted, default is no legend.
$\dots$	Additional arguments passed to the plot function. Use <code>colour</code> instead of <code>col</code> to create different colours.
$pm$	Logical scalar to enable pm.

## 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

	A 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

Dietze E, Hartmann K, Diekmann B, Ijmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

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

## perform the test and show q.min
L <- test.L(X = X.artificial, lw = lw, c = 100, plot = TRUE)
L$q.min

## a visualisation with more plot parameters
L <- test.L(X = X.artificial, lw = lw, 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)}
```

---

test.lw

---

*Function to test maximum valid lw value.*


---

**Description**

This function performs the weight transformation of the data matrix after Klován & 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.lw(X, lw)
```

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
lw	Numeric vector specifying the weight transformation limit, i.e. quantile; default is 0.

**Value**

A list with objects

step	Numeric scalar with position of last valid value.
lw.max	Numeric scalar with last valid value of lw.

**Author(s)**

Michael Dietze, Elisabeth Dietze

**References**

Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

Klovan JE, Imbrie J. 1971. An Algorithm and FORTRAN-IV Program for Large-Scale Q-Mode Factor Analysis and Calculation of Factor Scores. *Mathematical Geology* 3: 61-77.

**See Also**

[EMMA](#), [check.data](#), [test.parameters](#)

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

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

## test the vector
test.lw(X = X.artificial, lw = lw)
```

---

test.parameters	<i>Function to evaluate influence of model parameters.</i>
-----------------	--

---

**Description**

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

**Usage**

```
test.parameters(X, q, lw, c, rotation = "Varimax", plot, legend,
  ..., pm = FALSE)
```

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
q	Numeric vector of length two, specifying the minimum and maximum number of end-members to be modelled.
lw	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" (cf. Dietze et al., 2012). One out of the rotations provided in GPArotation is possible (cf. <a href="#">rotations</a> ).
plot	Character scalar, optional graphical output of the results. 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), "ol" (number of overlapping end-members). All plots except "ol" are colour-coded bitmaps of q, lw and the specified test parameter and line-plots the specified parameter vs. q.
legend	Character scalar, specifying legend position (cf. <a href="#">legend</a> ). If omitted, no legend will be plotted, default is no legend.
...	Additional arguments passed to the plot function. Since the function returns two plots (except for plot 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.
pm	Logical scalar to enable pm.

### Details

The mean total explained variance mRt 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 mRt maximum.

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

### Value

A list with result objects

mEm	Absolute row-wise model error.
mEn	Absolute column-wise model error.
mRm	Mean row-wise explained variance.
mRn	Mean column-wise explained variance.
mRt	Mean total explained variance.
ol	Number of overlapping end-member loadings.
q.max	Maximum number of meaningful end-members.

### Author(s)

Michael Dietze, Elisabeth Dietze

## References

Dietze E, Hartmann K, Diekmann B, IJmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

## See Also

[EMMA](#)

## Examples

```
## load example data set
data(X.artificial, envir = environment())

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

## define test parameters
q <- 2:8 # number of end-members
lw <- 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, lw = lw, plot = "mRt",
                     legend = "bottomright", cex = 0.7,
                     colour = rgb((1:7) / 7, 0.9, 0.2, 1))

## show maximum number of end-members
TP$q.max
```

---

test.robustness      *Function to test model robustness.*

---

## Description

All possible combinations of number of end-members and weight transformation limits are used to perform EMMA. The resulting loadings are written to an output matrix and mode positions (i.e. class with maximum loading) of all loadings are evaluated and returned.

## Usage

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

**Arguments**

X	Numeric matrix with m samples (rows) and n variables (columns).
q	Numeric vector with number of end-members to be modelled.
lw	Numeric vector specifying the weight transformation limits, i.e. quantiles; default is 0.
P	Numeric matrix, optional alternative input parameters for q and lw, either of the form m:3 with m variations in the columns q.min, q.max, lw or of the form m:2 with m variations in the columns q, lw.
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. <a href="#">rotations</a> ).
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.
...	Additional arguments passed to the plot function. 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.
pm	Logical scalar to enable pm.

**Details**

The function value \$loadings is redundant but was added for user convenience.

**Value**

A list with objects

q	Vector with q.
lw	Vector with lw.
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**

Dietze E, Hartmann K, Diekmann B, Ijmker J, Lehmkuhl F, Opitz S, Stauch G, Wuennemann B, Borchers A. 2012. An end-member algorithm for deciphering modern detrital processes from lake sediments of Lake Donggi Cona, NE Tibetan Plateau, China. *Sedimentary Geology* 243-244: 169-180.

**Examples**

```
## load example data set
data(X.artificial, envir = environment())

## Example 1 - perform the most simple test
q <- 4:7
lw <- seq(from = 0, to = 0.1, by = 0.02)

M1 <- test.robustness(X = X.artificial, q = q, lw = lw,
  ol.rej = 1, mRt.rej = 0.8,
  plot = TRUE,
  colour = c(4, 7),
  xlab = c(expression(paste("Grain size (", phi, ")"),
    sep = "")),
    expression(paste("Grain size (", phi, ")"),
    sep = "")))

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

```

---

X.artificial

*example data*


---

## Description

Artificial data set created by randomly mixed normal distributions

## Usage

```
data(X.artificial)
```

## Format

The format is: num [1:80, 1:80] 0.00032 0.00057 0.00037 0.00029 ...

## Details

The dataset is the result of five randomly mixed normal distributions, forming 80 samples, each represented by 80 classes. Some white noise is added.

## Examples

```

## load example data set
data(X.artificial)

## plot first 10 samples stacked in one line plot
plot(NA, xlim = c(1, 80), ylim = c(1, 14))
for(i in 1:10) {lines(1:80, X.artificial[i,] * 50 + i)}

```

# Index

## \*Topic **EMMA**

- check.data, [2](#)
- convert.units, [4](#)
- create.EM, [5](#)
- define.limits, [6](#)
- EMMA, [7](#)
- interpolate.classes, [9](#)
- mix.EM, [11](#)
- Mqs.uncertainty, [12](#)
- residual.EM, [15](#)
- robust.EM, [16](#)
- test.L, [18](#)
- test.lw, [20](#)
- test.parameters, [21](#)
- test.robustness, [23](#)

## \*Topic **datasets**

- X.artificial, [26](#)

## \*Topic **package**

- EMMAgeo-package, [2](#)

approx, [10](#)

check.data, [2](#), [21](#)

convert.units, [4](#)

create.EM, [5](#), [12](#)

define.limits, [6](#), [18](#)

eigen, [9](#)

EMMA, [3](#), [7](#), [10](#), [15](#), [18](#), [21](#), [23](#)

EMMAgeo (EMMAgeo-package), [2](#)

EMMAgeo-package, [2](#)

interpolate.classes, [4](#), [9](#)

legend, [8](#), [17](#), [19](#), [22](#)

mix.EM, [5](#), [11](#)

Mqs.uncertainty, [12](#)

npls, [9](#)

residual.EM, [15](#)

robust.EM, [7](#), [15](#), [16](#)

rotations, [8](#), [9](#), [13](#), [22](#), [24](#)

spline, [10](#)

test.L, [18](#)

test.lw, [20](#)

test.parameters, [9](#), [14](#), [21](#), [21](#)

test.robustness, [6](#), [7](#), [14](#), [18](#), [23](#)

X.artificial, [26](#)