End-member modelling analysis
The R-package EMMAgeo

Elisabeth Dietze[1], Michael Dietze[2]

1 - GFZ German Research Centre for Geosciences, Section 5.2 Climate Dynamics and Landscape Evolution
2 - GFZ German Research Centre for Geosciences, Section 5.1 Geomorphology

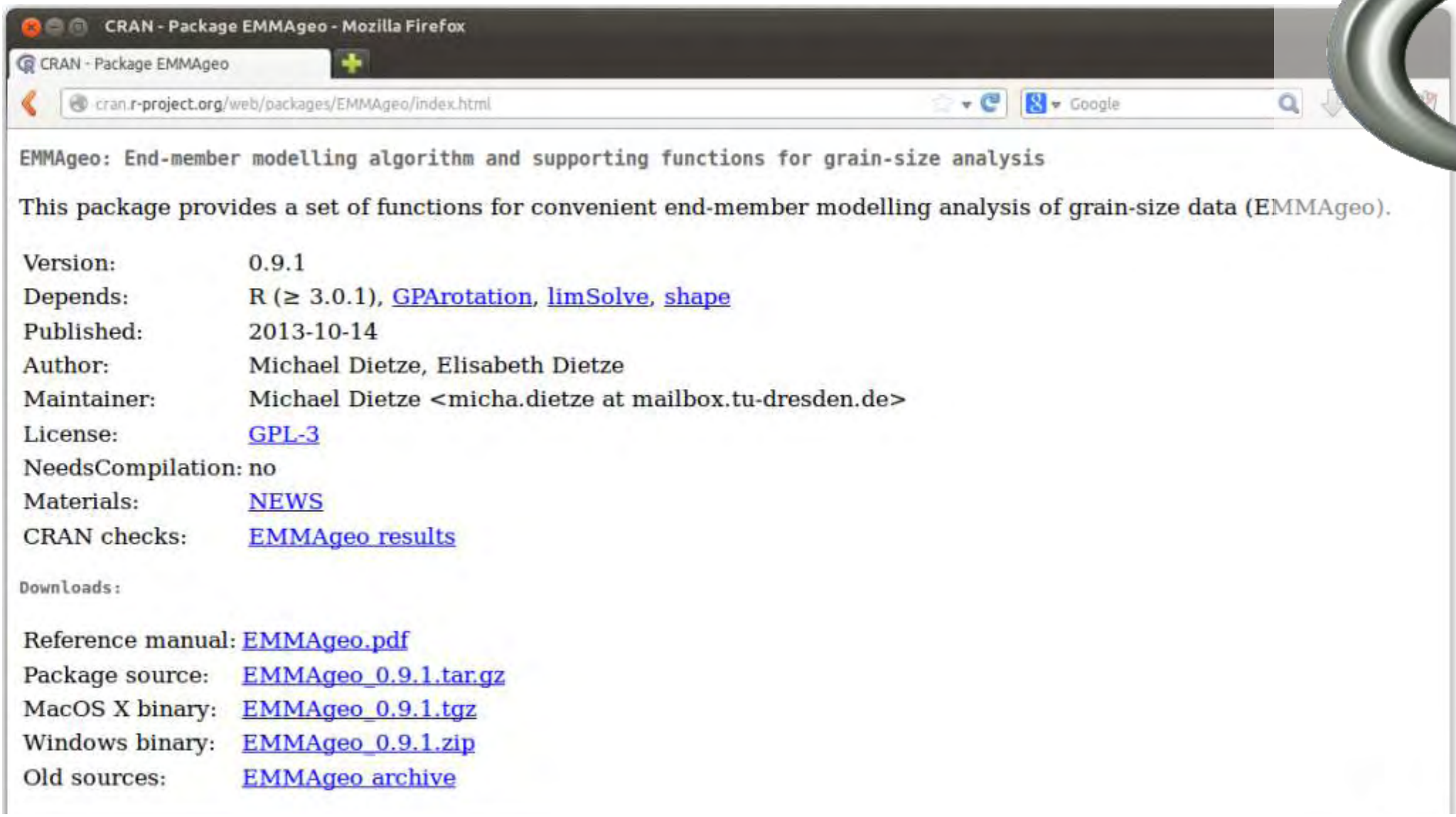
End-member modelling analysis

The R-package EMMAgeo

Michael Dietze

GFZ German Research Centre for Geosciences, Section 5.1 Geomorphology

EMMAgeo - on CRAN

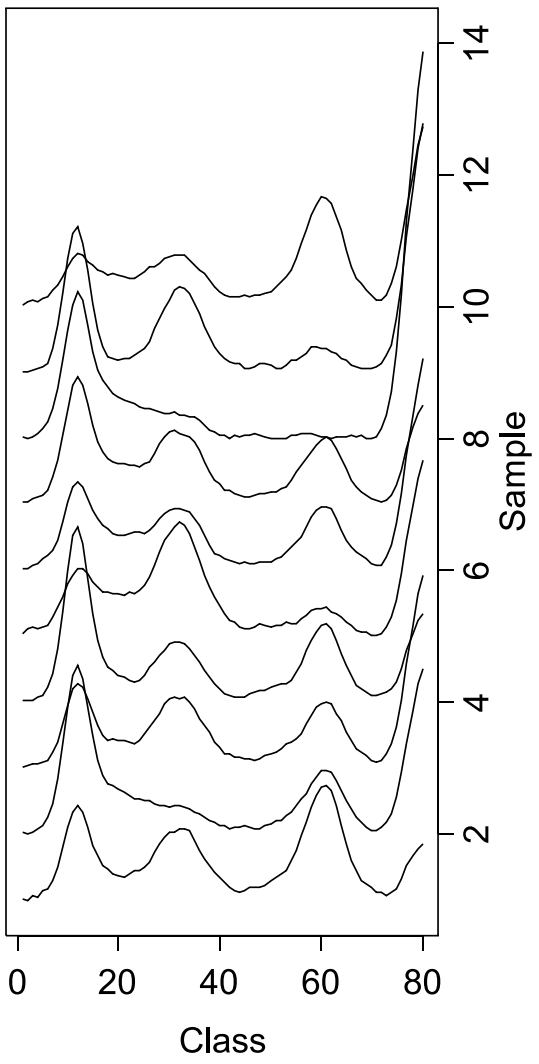


EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

Example data set



Documentation (& Tutorial)

Package 'EMMAgeo'

October 14, 2013

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:

Rescale data matrix X to constant sum c.	`X  <- X / apply(X, 1, sum) * c`

Weight tranformation after Miesch (1970).	`qts <- function(X, lw) quantile(X, c(lw, 1-lw), type = 5)`
	`ls <- t(apply(X, 2, qts, lw = lw))`
	`W <- t((t(X) - ls[,1]) / (ls[,2] - ls[,1]))`

Similarity matrix calculation (major product).	`A <- t(W) %*% W`

Eigen space extraction.	`EIG <- eigen(A)`
	
	`V <- EIG$vectors[,order(seq(ncol(A), 1, -1))]`
	`Vf <- V[,order(seq(ncol(A), 1, -1))]`
	`L <- EIG$values[order(seq(ncol(A), 1, -1))]`
	`Lv <- cumsum(sort(L / sum(L), decreasing = TRUE))`

Varimax rotation of the eigen vector matrix Vf.	`Vr <- do.call(rotation, list(Vf[,1:q]))`

Extract and sort (decreasing) factor loadings and write them to matrix Vq.
Rescale (Vqr) and normalise (Vqn) the factor loadings column-wise.
	`Vq <- Vr$loadings[,order(seq(q, 1, -1))]`
	`Vqr <- t(t(Vq) / apply(Vq, 2, sum)) * c`
	`Vqr <- t(Vqr)`
	`Vqn <- t((Vqr - apply(Vqr, 1, min)) / (`
	`  apply(Vqr, 1, max) - apply(Vqr, 1, min)))`

Calculate factor scores matrix (Mq) by non-negative least square fitting of Vqn and transposed row-wise weight-transformed data W.	`Mq <- matrix(nrow = nrow(X), ncol = q)` `for (i in 1:nrow(X)) {Mq[i,] = nnls(Vqn, as.vector(t(W[i,])))$X}`
Model the dataset (Wm) as the minor product	`Wm <- Mq %*% t(Vqn)`

Rescale the factor loadings matrix Vqn to Vqsn.	`s <- (c - sum(ls[,1])) / apply(Vqn * unname(ls[,2] - ls[,1]), 2, sum)`
	`Vqs <- Vqn`
	`for(i in 1:q) {Vqs[,i] <- t(s[i] * t(Vqn[,i]) * (ls[,2] - ls[,1]) + ls[,1])}`
	`Vqsn <- t(t(Vqs)/ apply(Vqs, 2, sum)) * c`

Rescale factor scores (Mq) to matrix Mqs and calculate variance explanied by scores.	`Mqs <- t(t(Mq) / s) / apply(t(t(Mq) / s), 1, sum)` `Mqs.var <- diag(var(Mqs)) / sum(diag(var(Mqs))) * 100`

Evaluate measures of model goodness.	`Em <- as.vector(apply(X - Xm, 1, mean))`
	`En <- as.vector(apply(X - Xm, 2, mean))`
	`Rm <- diag(cor(t(X), t(Xm))^2)`
	`Rn <- diag(cor(X, Xm)^2)`

Principal component analysis

Rescale data matrix X to constant sum c.	`X  <- X / apply(X, 1, sum) * c`

Weight tranformation after Miesch (1970).	`qts <- function(X, lw) quantile(X, c(lw, 1-lw), type = 5)`
	`ls <- t(apply(X, 2, qts, lw = lw))`
	`W <- t((t(X) - ls[,1]) / (ls[,2] - ls[,1]))`

Similarity matrix calculation (major product).	`A <- t(W) %*% W`

Eigen space extraction.	`EIG <- eigen(A)`
	
	`V <- EIG$vectors[,order(seq(ncol(A), 1, -1))]`
	`Vf <- V[,order(seq(ncol(A), 1, -1))]`
	`L <- EIG$values[order(seq(ncol(A), 1, -1))]`
	`Lv <- cumsum(sort(L / sum(L), decreasing = TRUE))`

Varimax rotation of the eigen vector matrix Vf.	`Vr <- do.call(rotation, list(Vf[,1:q]))`

Extract and sort (decreasing) factor loadings and write them to matrix Vq.
Rescale (Vqr) and normalise (Vqn) the factor loadings column-wise.
	`Vq <- Vr$loadings[,order(seq(q, 1, -1))]`
	`Vqr <- t(t(Vq) / apply(Vq, 2, sum)) * c`
	`Vqr <- t(Vqr)`
	`Vqn <- t((Vqr - apply(Vqr, 1, min)) / (`
	`  apply(Vqr, 1, max) - apply(Vqr, 1, min)))`

Calculate factor scores matrix (Mq) by non-negative least square fitting of Vqn and transposed row-wise weight-transformed data W.	`Mq <- matrix(nrow = nrow(X), ncol = q)` `for (i in 1:nrow(X)) {Mq[i,] = nnls(Vqn, as.vector(t(W[i,])))$X}`
Model the dataset (Wm) as the minor product	`Wm <- Mq %*% t(Vqn)`

Rescale the factor loadings matrix Vqn to Vqsn.	`s <- (c - sum(ls[,1])) / apply(Vqn * unname(ls[,2] - ls[,1]), 2, sum)`
	`Vqs <- Vqn`
	`for(i in 1:q) {Vqs[,i] <- t(s[i] * t(Vqn[,i]) * (ls[,2] - ls[,1]) + ls[,1])}`
	`Vqsn <- t(t(Vqs)/ apply(Vqs, 2, sum)) * c`

Rescale factor scores (Mq) to matrix Mqs and calculate variance explanied by scores.	`Mqs <- t(t(Mq) / s) / apply(t(t(Mq) / s), 1, sum)` `Mqs.var <- diag(var(Mqs)) / sum(diag(var(Mqs))) * 100`

Evaluate measures of model goodness.	`Em <- as.vector(apply(X - Xm, 1, mean))`
	`En <- as.vector(apply(X - Xm, 2, mean))`
	`Rm <- diag(cor(t(X), t(Xm))^2)`
	`Rn <- diag(cor(X, Xm)^2)`

Rescale data matrix X to constant sum c.

```
X  <- X / apply(X, 1, sum) * c
```

Weight tranformation after Miesch (1970).

```
qts <- function(X, lw) quantile(X, c(lw, 1-lw), type = 5)
ls <- t(apply(X, 2, qts, lw = lw))
W <- t((t(X) - ls[,1]) / (ls[,2] - ls[,1]))
```

Similarity matrix calculation (major product).

```
A <- t(W) %*% W
```

Eigen space extraction.

```
EIG <- eigen(A)

V <- EIG$vectors[,order(seq(ncol(A), 1, -1))]
Vf <- V[,order(seq(ncol(A), 1, -1))]
L <- EIG$values[order(seq(ncol(A), 1, -1))]
Lv <- cumsum(sort(L / sum(L), decreasing = TRUE))
```

Principal component analysis

Varimax rotation of the eigen vector matrix Vf.

```
Vr <- do.call(rotation, list(Vf[,1:q]))
```

Extract and sort (decreasing) factor loadings and write them to matrix Vq.
Rescale (Vqr) and normalise (Vqn) the factor loadings column-wise.

```
Vq <- Vr$loadings[,order(seq(q, 1, -1))]
Vqr <- t(t(Vq) / apply(Vq, 2, sum)) * c
Vqr <- t(Vqr)
Vqn <- t((Vqr - apply(Vqr, 1, min)) / (
  apply(Vqr, 1, max) - apply(Vqr, 1, min)))
```

Calculate factor scores matrix (Mq) by non-negative least square fitting of Vqn and transposed row-wise weight-transformed data W.

```
Mq <- matrix(nrow = nrow(X), ncol = q)
for (i in 1:nrow(X)) {Mq[i,] = nnls(Vqn, as.vector(t(W[i,])))$X}
```

Model the dataset (Wm) as the minor product

```
Wm <- Mq %*% t(Vqn)
```

Factor analysis

Rescale the factor loadings matrix Vqn to Vqsn.

```
s <- (c - sum(ls[,1])) / apply(Vqn * unname(ls[,2] - ls[,1]), 2, sum)
Vqs <- Vqn
for(i in 1:q) {Vqs[,i] <- t(s[i] * t(Vqn[,i]) * (ls[,2] - ls[,1]) + ls[,1])}
Vqsn <- t(t(Vqs)/ apply(Vqs, 2, sum)) * c
```

Rescale factor scores (Mq) to matrix Mqs and calculate variance explanied by scores.

```
Mqs <- t(t(Mq) / s) / apply(t(t(Mq) / s), 1, sum)
Mqs.var <- diag(var(Mqs)) / sum(diag(var(Mqs))) * 100
```

Evaluate measures of model goodness.

```
Em <- as.vector(apply(X - Xm, 1, mean))
En <- as.vector(apply(X - Xm, 2, mean))
Rm <- diag(cor(t(X), t(Xm))^2)
Rn <- diag(cor(X, Xm)^2)
```

Rescale data matrix X to constant sum c.

```
X  <- X / apply(X, 1, sum) * c
```

Principal component analysis

Weight tranformation after Miesch (1970).

```
qts <- function(X, lw) quantile(X, c(lw, 1-lw), type = 5)
ls <- t(apply(X, 2, qts, lw = lw))
W <- t((t(X) - ls[,1]) / (ls[,2] - ls[,1]))
```

Similarity matrix calculation (major product).

```
A <- t(W) %*% W
```

Eigen space extraction.

```
EIG <- eigen(A)

V <- EIG$vectors[,order(seq(ncol(A), 1, -1))]
Vf <- V[,order(seq(ncol(A), 1, -1))]
L <- EIG$values[order(seq(ncol(A), 1, -1))]
Lv <- cumsum(sort(L / sum(L), decreasing = TRUE))
```

Varimax rotation of the eigen vector matrix Vf.

```
Vr <- do.call(rotation, list(Vf[,1:q]))
```

Extract and sort (decreasing) factor loadings and write them to matrix Vq.
Rescale (Vqr) and normalise (Vqn) the factor loadings column-wise.

```
Vq <- Vr$loadings[,order(seq(q, 1, -1))]
Vqr <- t(t(Vq) / apply(Vq, 2, sum)) * c
Vqr <- t(Vqr)
Vqn <- t((Vqr - apply(Vqr, 1, min)) / (
  apply(Vqr, 1, max) - apply(Vqr, 1, min)))
```

Calculate factor scores matrix (Mq) by non-negative least square fitting of Vqn and transposed row-wise weight-transformed data W.

```
Mq <- matrix(nrow = nrow(X), ncol = q)
for (i in 1:nrow(X)) {Mq[i,] = nnls(Vqn, as.vector(t(W[i,])))$X}
```

Model the dataset (Wm) as the minor product

```
Wm <- Mq %*% t(Vqn)
```

Factor analysis

Rescale the factor loadings matrix Vqn to Vqsn.

```
s <- (c - sum(ls[,1])) / apply(Vqn * unname(ls[,2] - ls[,1]), 2, sum)
Vqs <- Vqn
for(i in 1:q) {Vqs[,i] <- t(s[i] * t(Vqn[,i]) * (ls[,2] - ls[,1]) + ls[,1])}
Vqsn <- t(t(Vqs)/ apply(Vqs, 2, sum)) * c
```

Rescale factor scores (Mq) to matrix Mqs and calculate variance explanied by scores.

```
Mqs <- t(t(Mq) / s) / apply(t(t(Mq) / s), 1, sum)
Mqs.var <- diag(var(Mqs)) / sum(diag(var(Mqs))) * 100
```

End-member modelling analysis

Evaluate measures of model goodness.

```
Em <- as.vector(apply(X - Xm, 1, mean))
En <- as.vector(apply(X - Xm, 2, mean))
Rm <- diag(cor(t(X), t(Xm))^2)
Rn <- diag(cor(X, Xm)^2)
```

EMMAgeo - package overview

Functions

- convert.units()
- interpolate.classes()
- check.data()
- test.lw()
- test.L()
- test.parameters()
- EMMA()
- test.robustness()
- robust.EM()
- residual.EM()
- Mqs.uncertainty()
- create.EM()
- mix.EM()

convert.units()

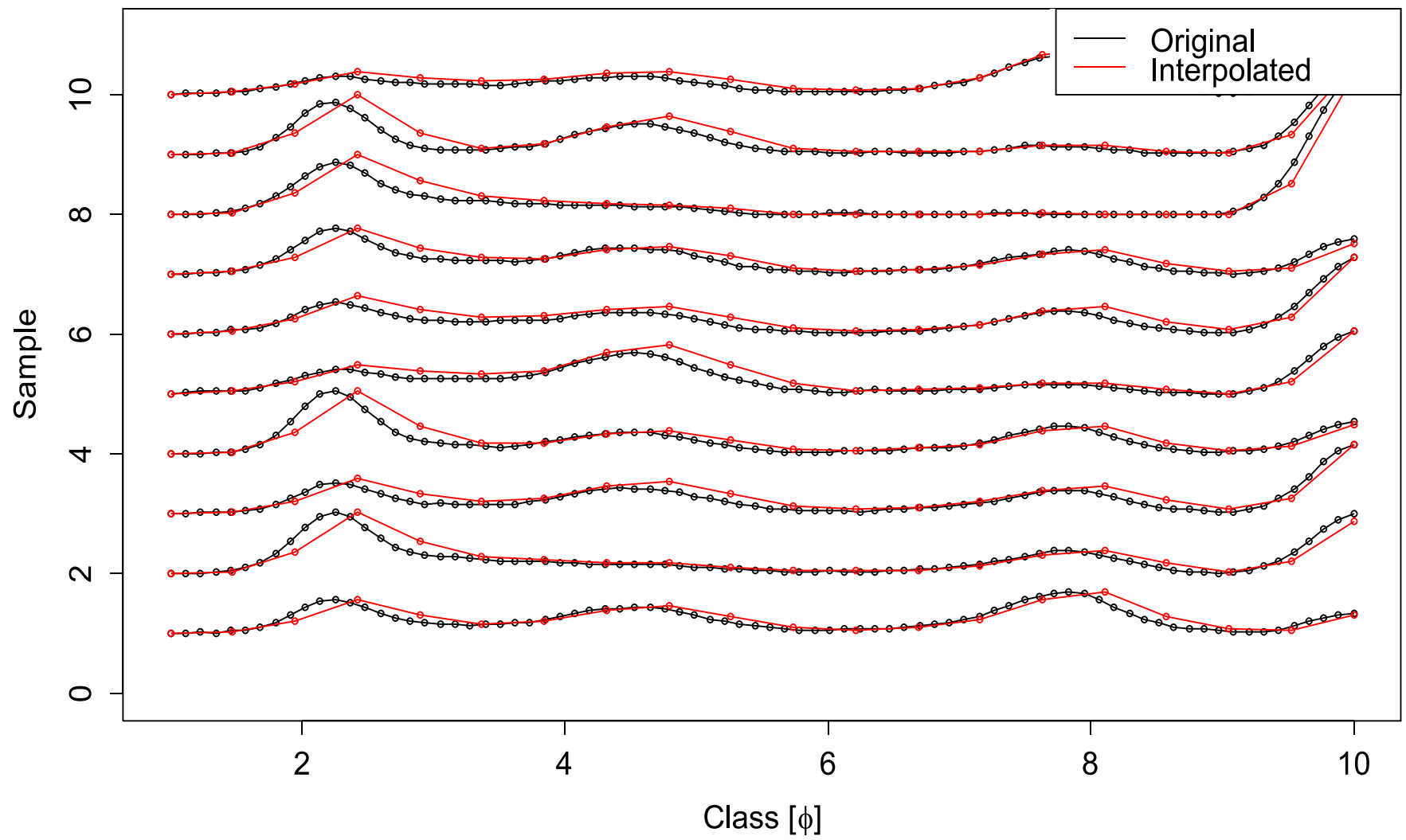
```
phi <- -2:5

mu  <- convert.units(phi = phi)

mu

[1] 4000.00 2000.00 1000.00  500.00  250.00  125.00   62.50   31.25

convert.units(mu = mu)

[1] -2 -1  0  1  2  3  4  5
```

EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

interpolate.classes()



EMMAgeo - package overview

Functions

- convert.units()
- interpolate.classes()
- **check.data()**
- test.lw()
- test.L()
- test.parameters()
- EMMA()
- test.robustness()
- robust.EM()
- residual.EM()
- Mqs.uncertainty()
- create.EM()
- mix.EM()

check.data()

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

[1] "Data matrix passed test... OK"

[2] "End-member vector passed test... OK"

[3] "Weight transformation limit vector
     passed test... OK"

[4] "Scaling parameter passed test... OK"

[5] "NA-test passed... OK"

[6] "Test for zero-only values passed... OK"

[7] "Maximum weight transformation limit
     value passed test... OK"

[8] "All samples sum up to constant sum... OK"
```

Input data must:

- be free of NA-values
- be of numeric data type
- contain no zero-only columns
- sum to constant value
- within stable limits (l_w)

EMMAgeo - package overview

Functions

- convert.units()
- interpolate.classes()
- check.data()
- **test.lw()**
- test.L()
- test.parameters()
- EMMA()
- test.robustness()
- robust.EM()
- residual.EM()
- Mqs.uncertainty()
- create.EM()
- mix.EM()

test.lw()

```
lw <- seq(from = 0, to = 0.5, by = 0.02)

test.lw(X = X.artificial, lw = lw)

$step
[1] 12

$lw.max
[1] 0.22
```

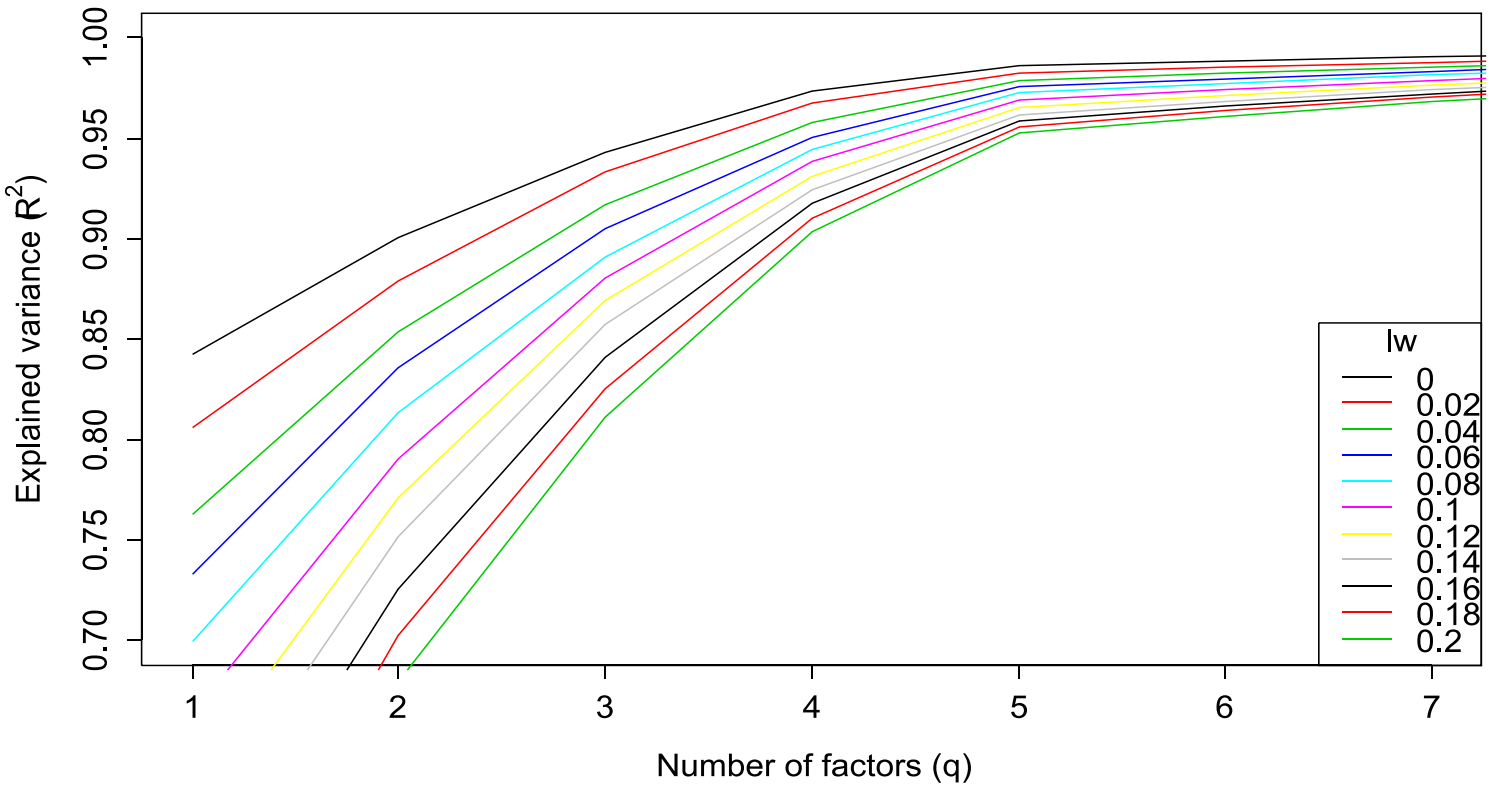
Weight transformation limits must be below a critical value from which on no similarity matrix can be computed any more. The function returns the last possible value for l_w.

EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

test.L()



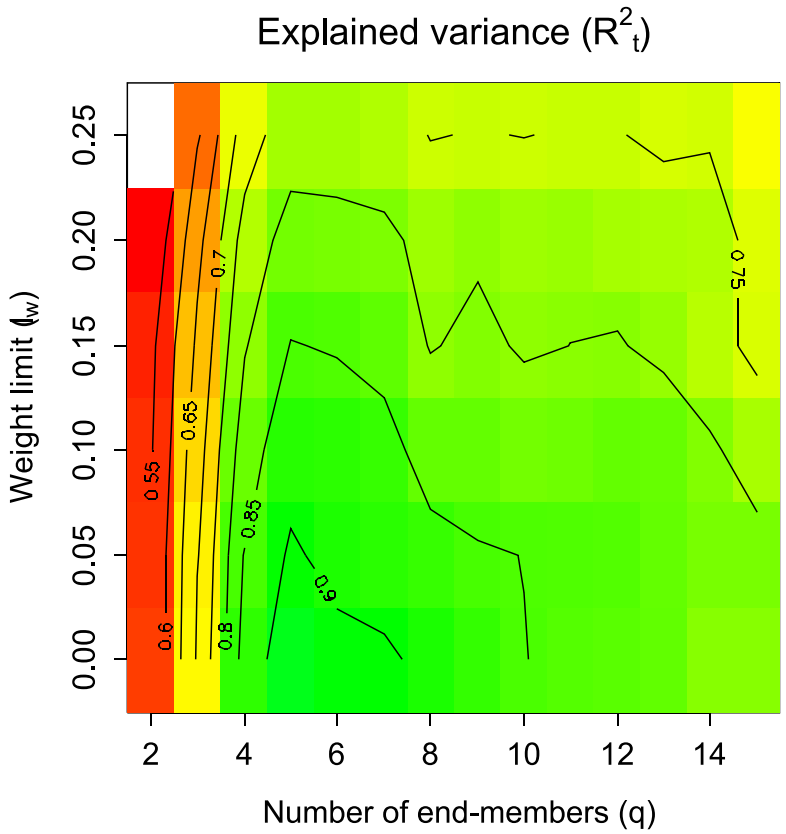

```
L$q.min

[1] 4 4 4 4 5 5 5 5 5 5 5
```

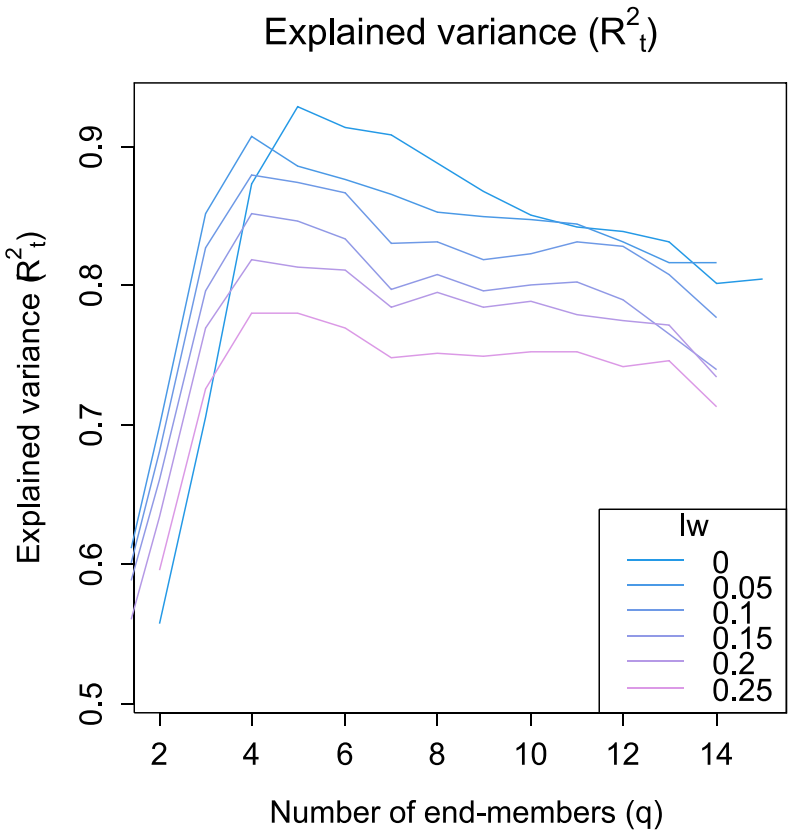
EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

test.parameters()






```
TP$q.max
```

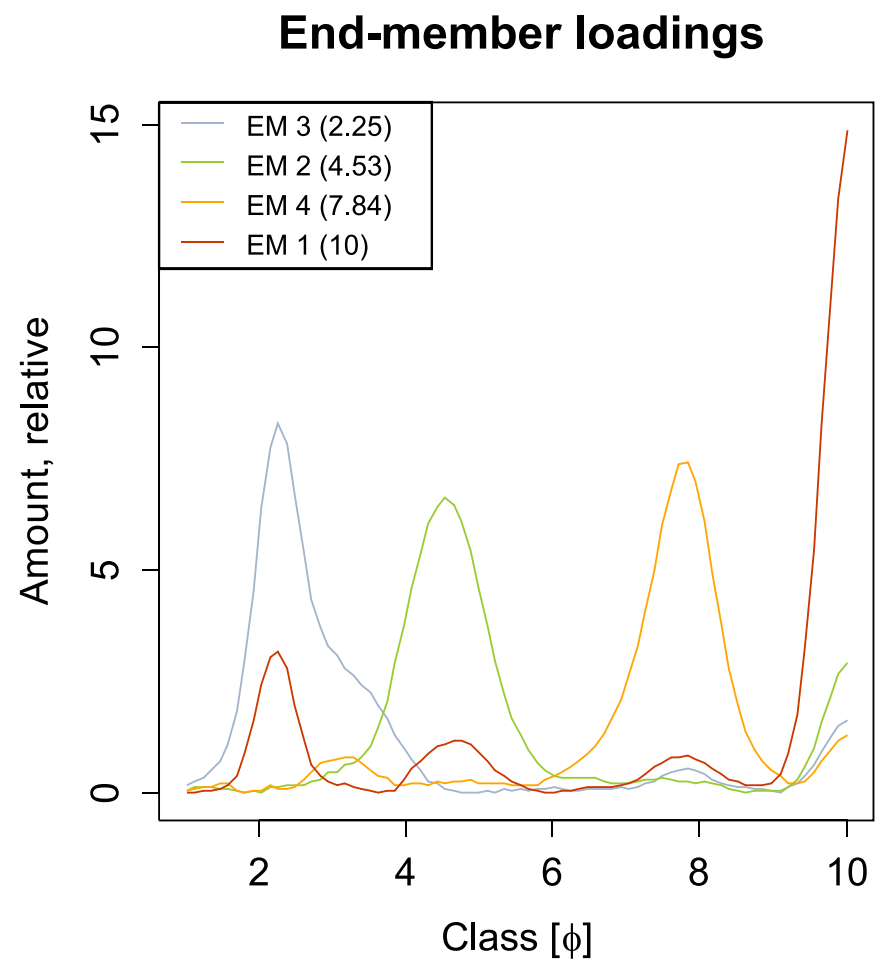
```
[1]  5  5  5  5  5 NA
```

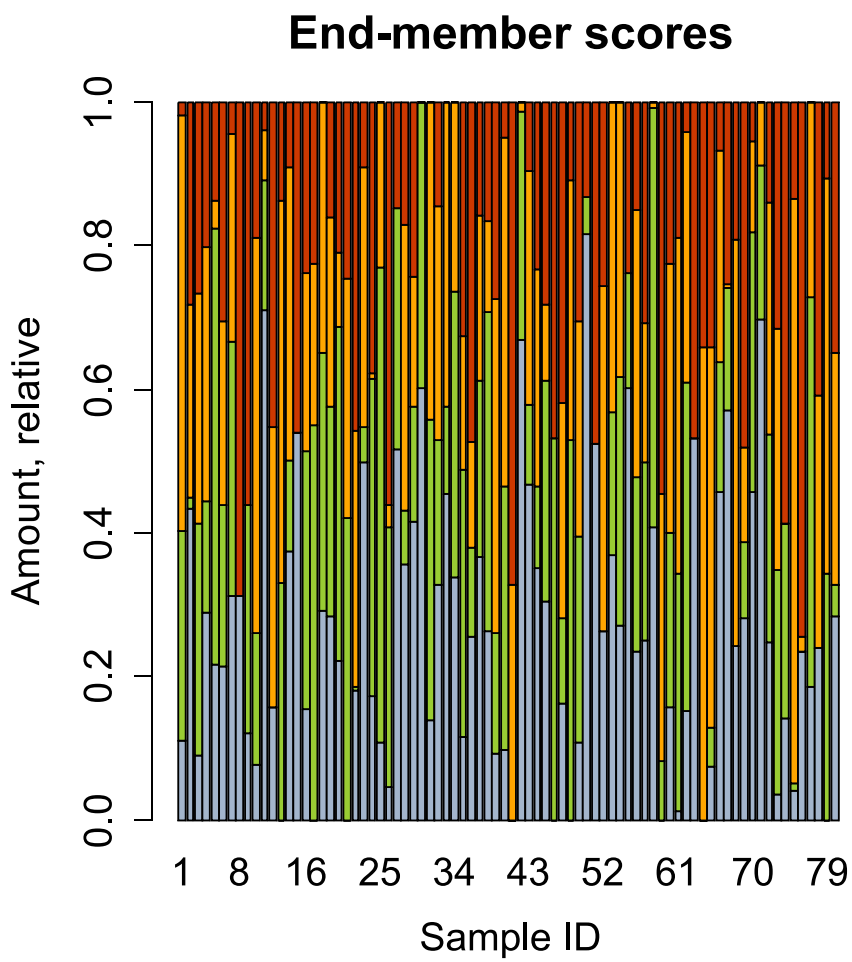
EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

EMMA()



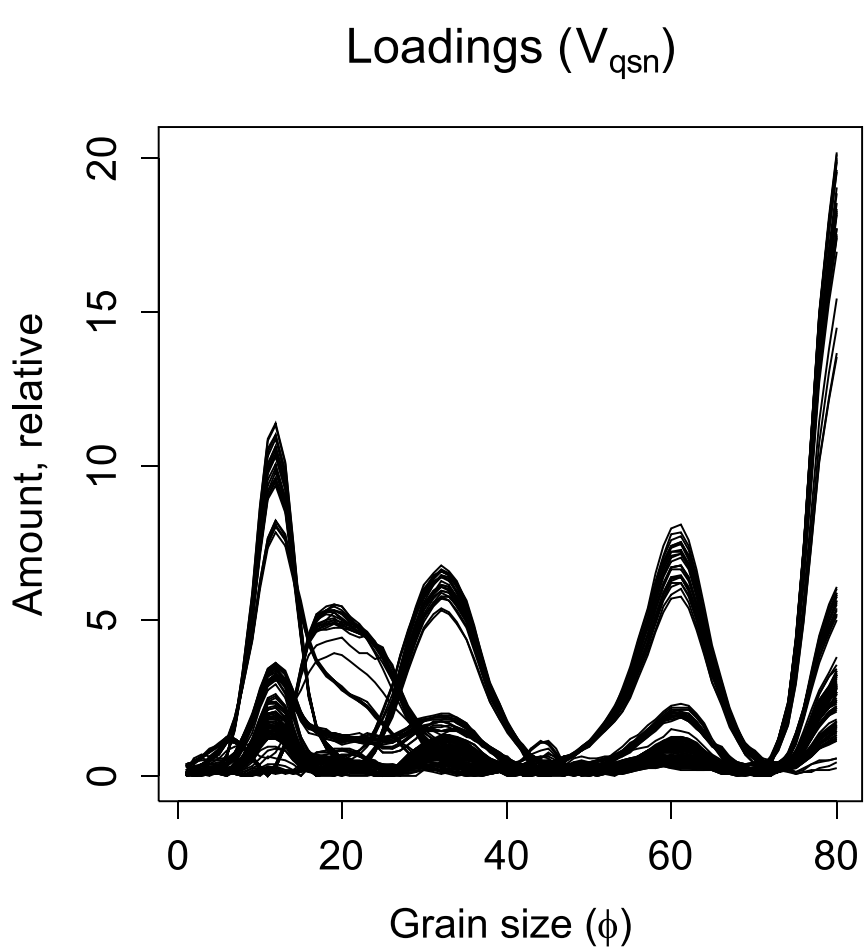


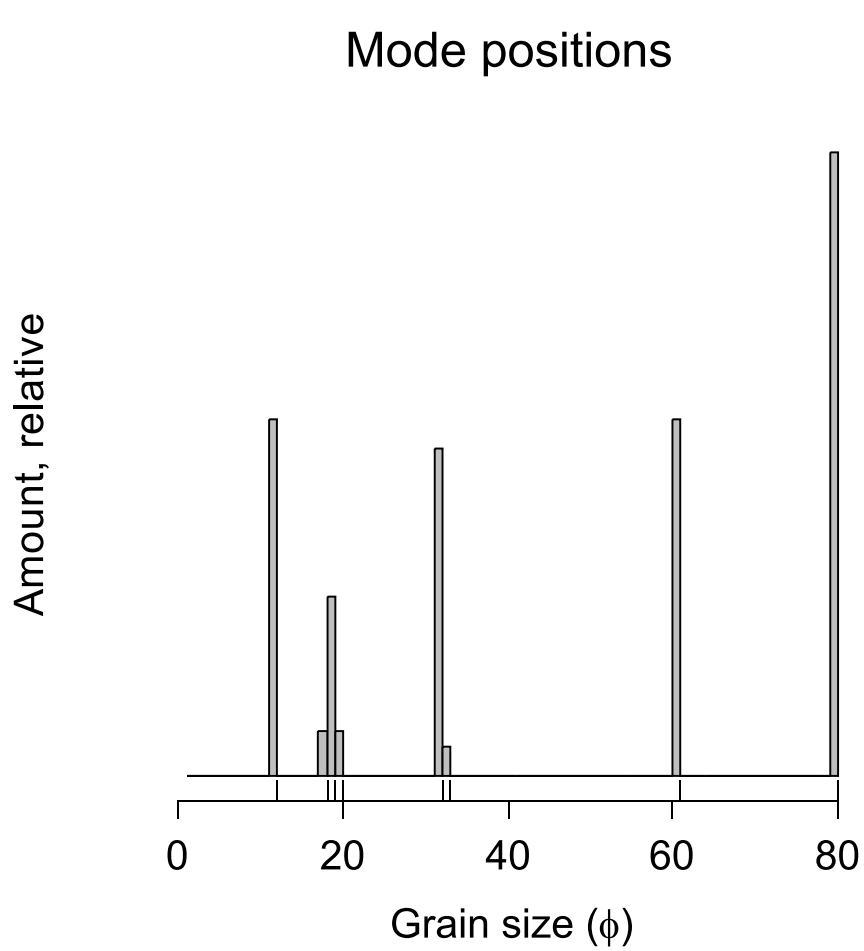
EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

test.robustness()



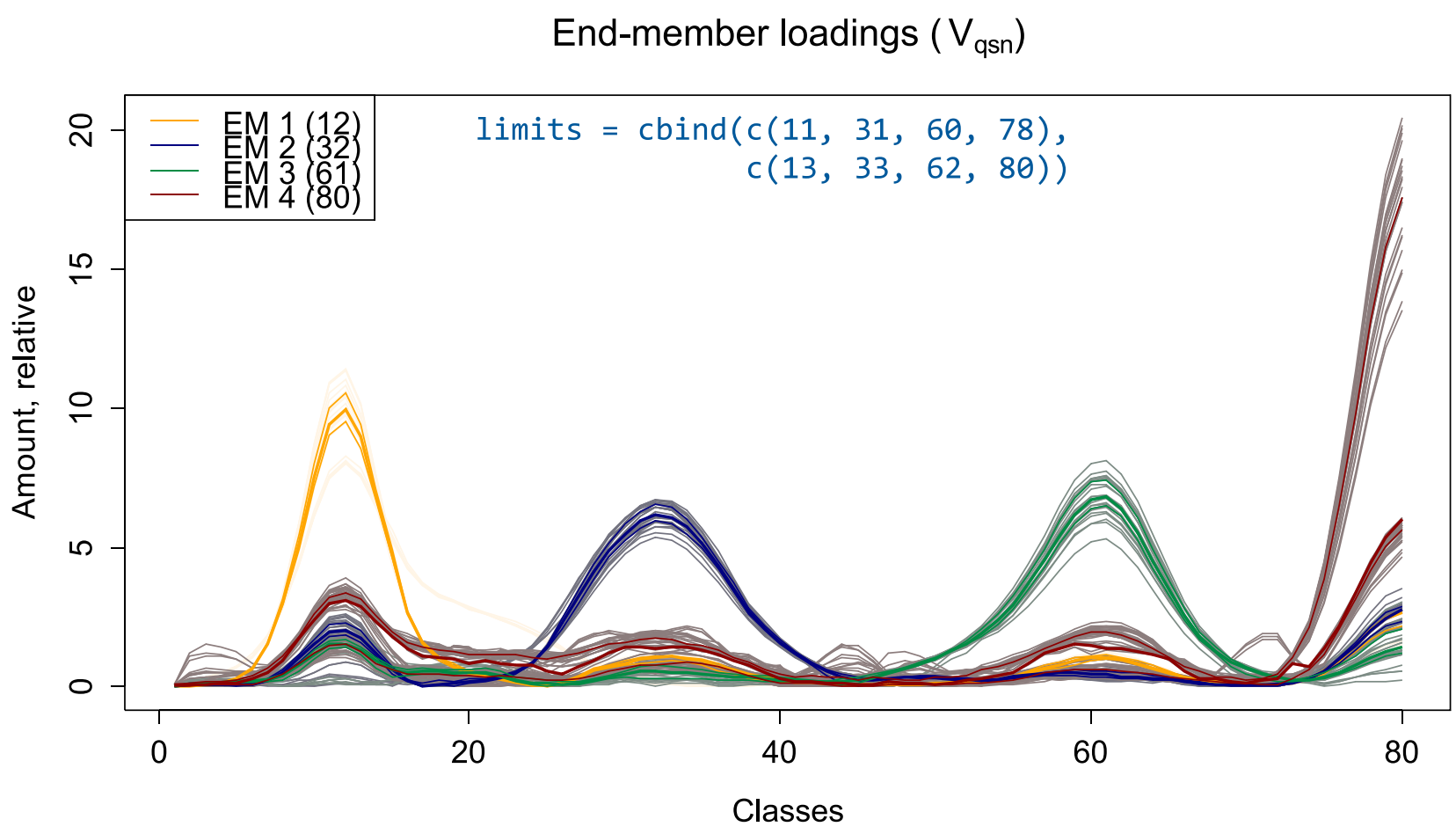


EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

test.robustness()

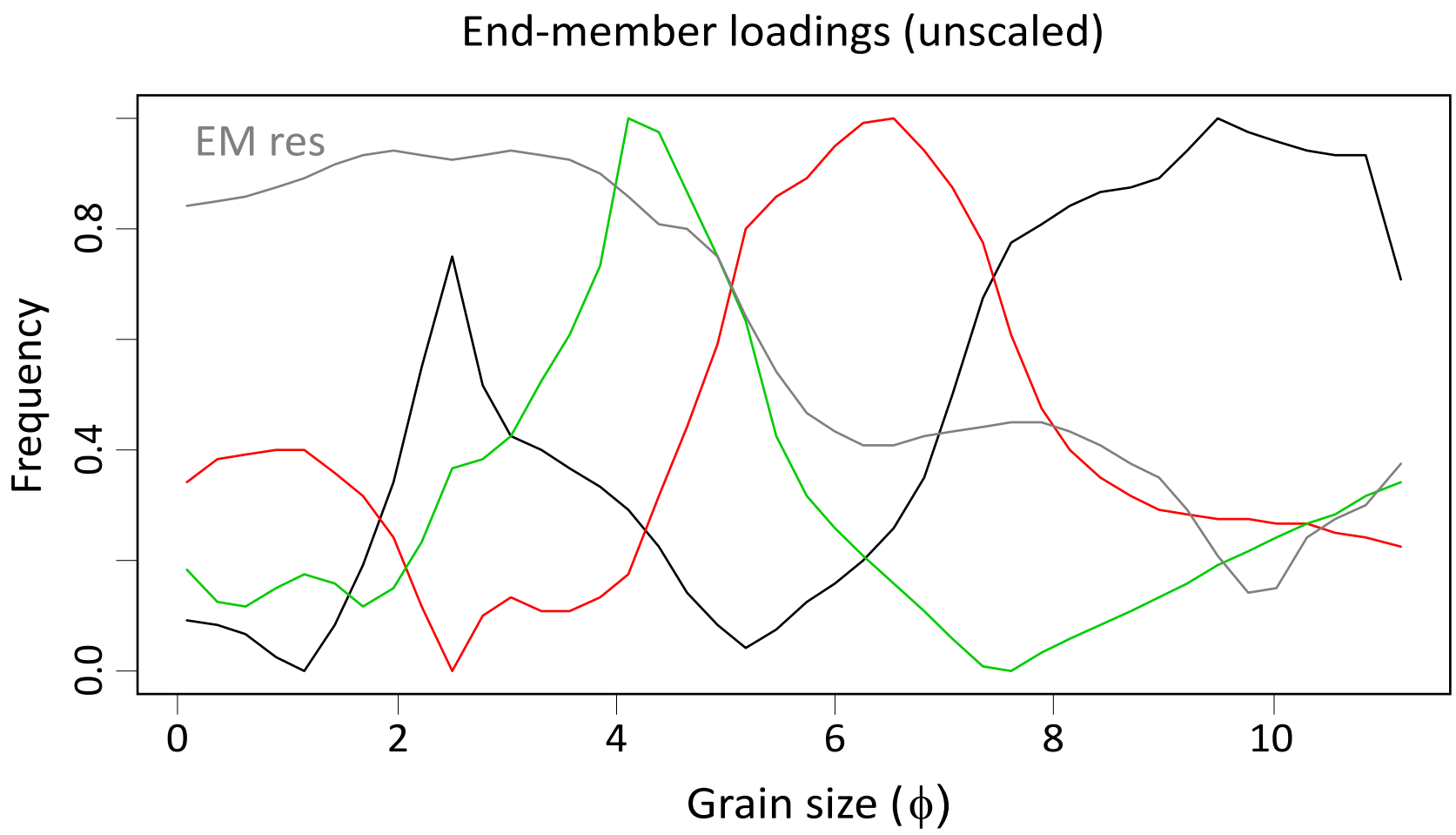


EMMAgeo - package overview

Functions

- convert.units()
- interpolate.classes()
- check.data()
- test.lw()
- test.L()
- test.parameters()
- EMMA()
- test.robustness()
- robust.EM()
- **residual.EM()**
- Mqs.uncertainty()
- create.EM()
- mix.EM()

residual.EM()

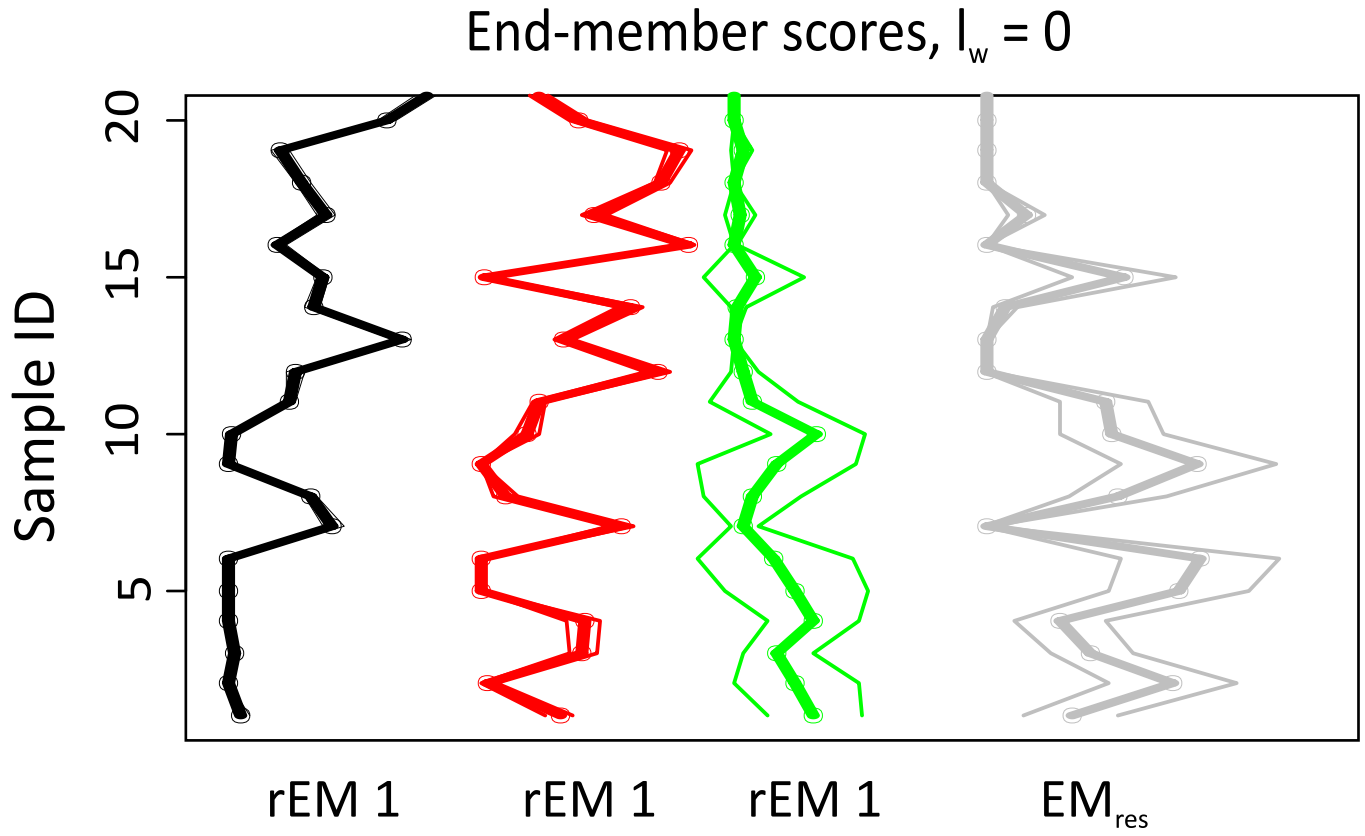


EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

Mqs.uncertainty()



EMMAgeo - package overview

Functions

```
convert.units()
interpolate.classes()
check.data()
test.lw()
test.L()
test.parameters()
EMMA()
test.robustness()
robust.EM()
residual.EM()
Mqs.uncertainty()
create.EM()
mix.EM()
```

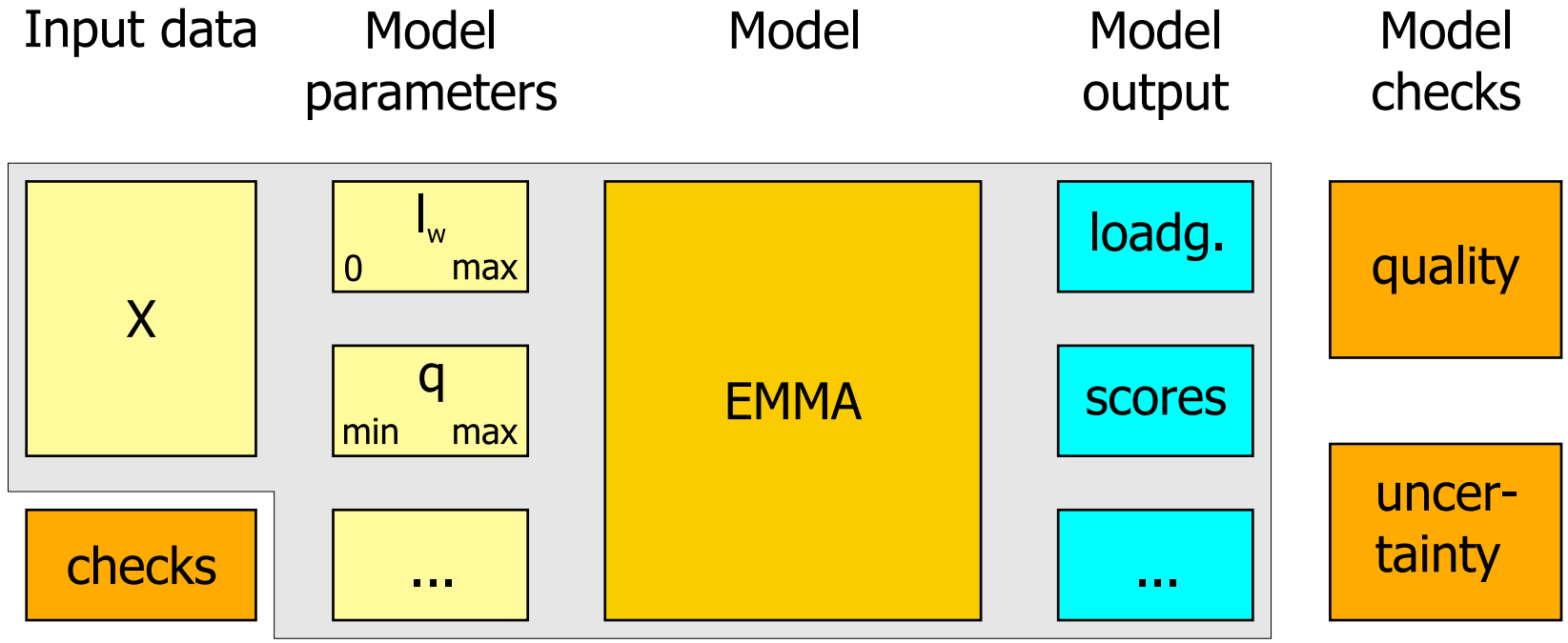
create.EM(), mix.EM()

Functions to create and mix user-defined end-members, e.g. for hypothesis testing.

Robust end-member modelling - a protocol suggestion

Protocol steps

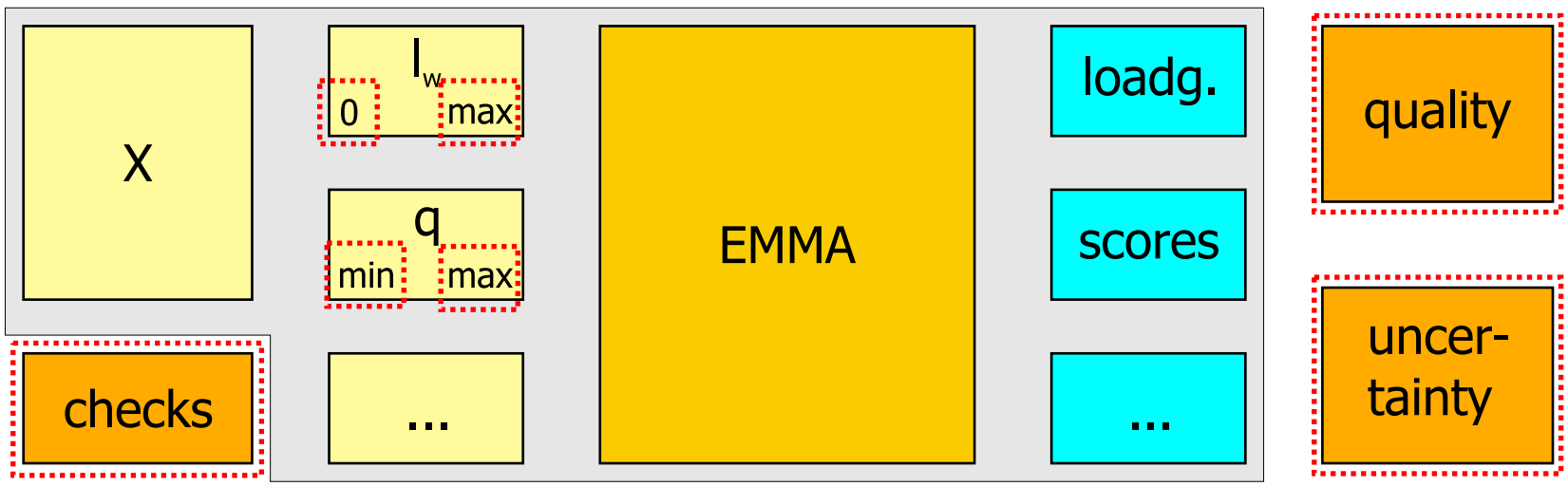
1. check the data
2. bracket weight transformation limits
3. bracket numbers of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty



Robust end-member modelling - a protocol suggestion

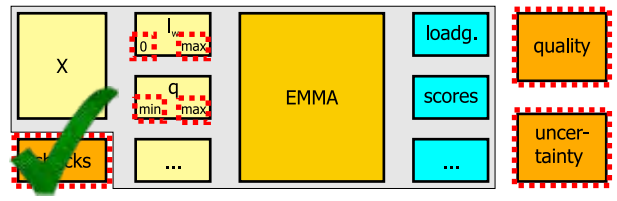
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket numbers of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty



Protocol is explicitly formulated as R-script and can be run straight forward and adapted if necessary.

Robust end-member modelling - a protocol suggestion



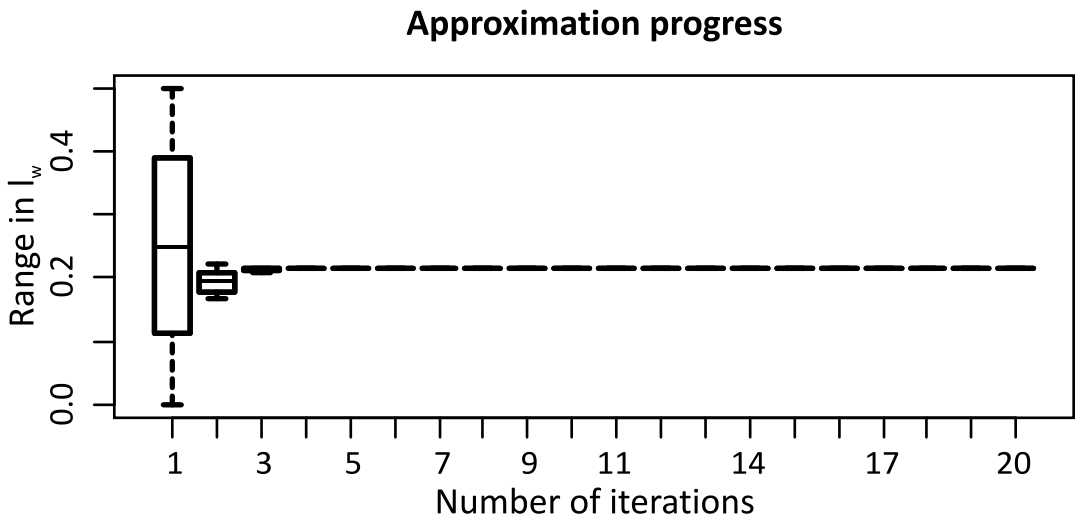
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

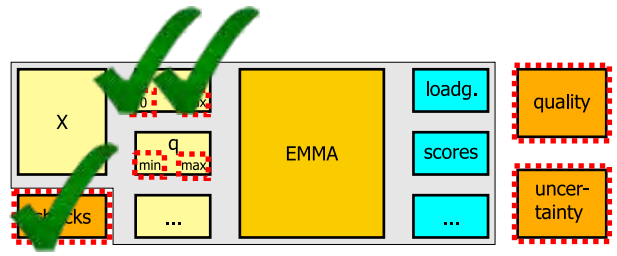
Functions: test.lw()

Goal is to find the highest possible weight transformation value l_w.
Lower limit is zero, i.e. no weight transformation.

```
for(i in 1:10) {
 if(i == 1) {lw.new <- seq(0,
                           0.5,
                           length = 10)}
 lw <- test.lw(X = X, lw = lw.new)
 lw.new <- seq(lw.new[lw$step],
               lw.new[lw$step + 1],
               length = 10)
}
lw.max <- lw$lw.max
lw.max
```



Robust end-member modelling - a protocol suggestion



Protocol steps

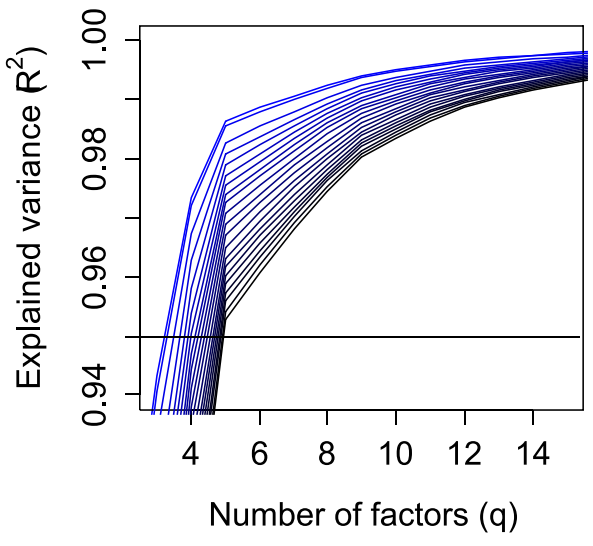
1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

Functions: test.L(), test.parameters()

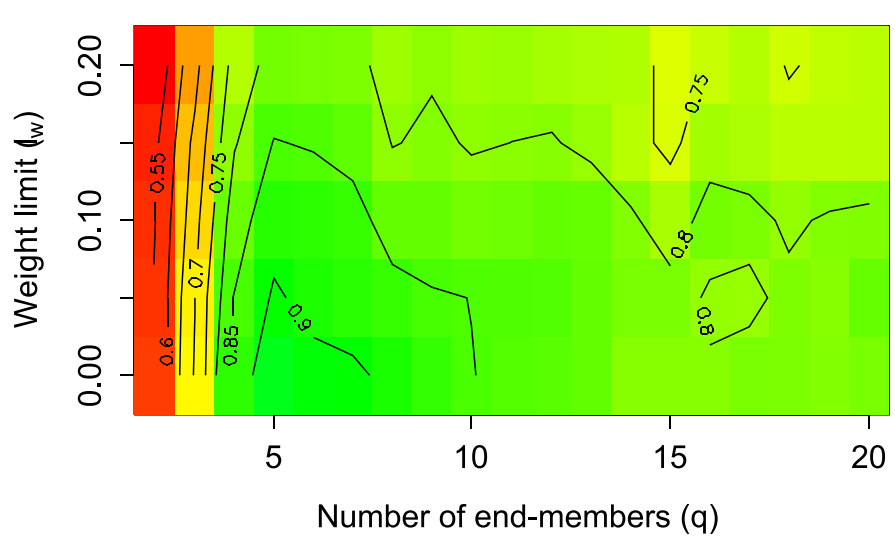
Goal is to find the minimum and maximum number of eigenvectors to adequately model the data set.

Mathematical definition of "adequate" may differ from geoscientific one.

Variance explained by factors



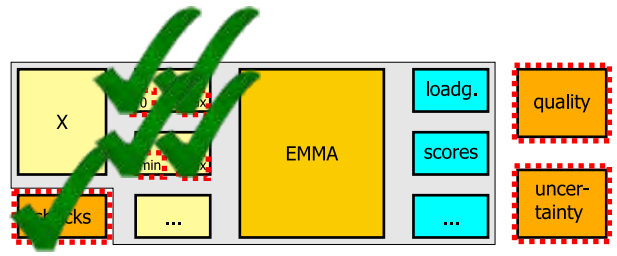
Explained variance ($\overline{R^2}_t$)




```
q.min <- test.L(X = X, lw = lw)$q.min
q.min
```

```
q.max <- test.parameters(X = X, q = q, lw = lw)$q.max
q.max
```

Robust end-member modelling - a protocol suggestion



Protocol steps

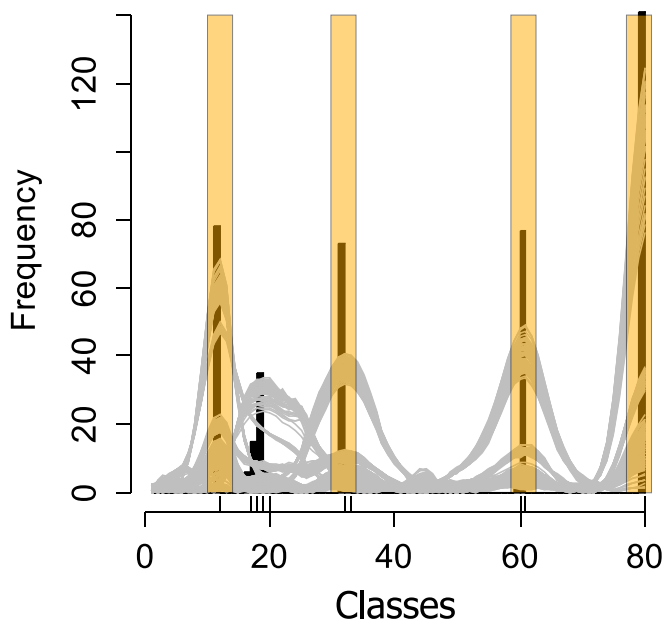
1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

Functions: test.robustness(), robust.EM()

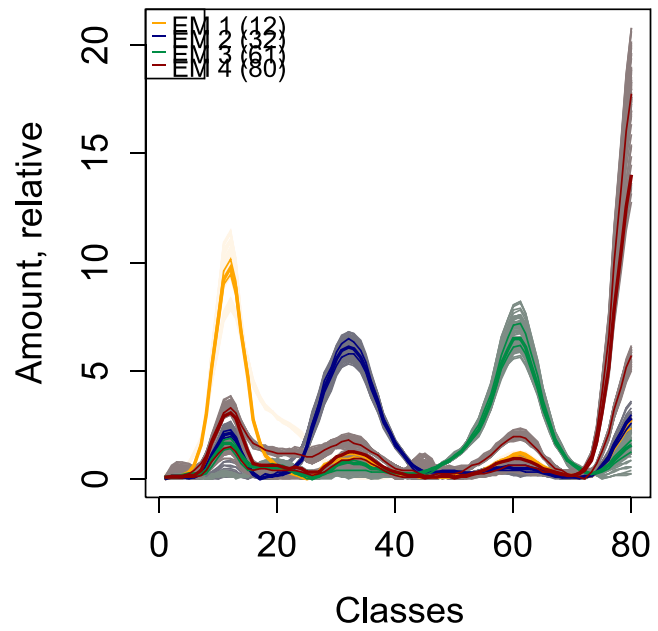
Several model scenarios are equally likely. Robust end-members persist throughout many scenarios.

Goal is to identify and isolate robust end-members.

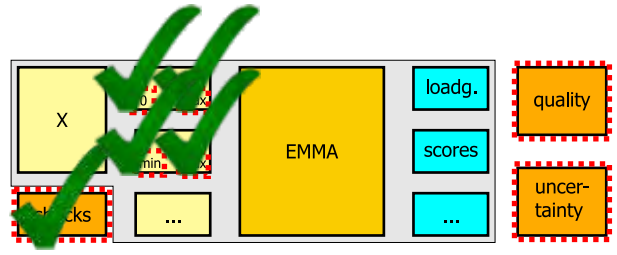
Mode positions of robust end-membe



End-member loadings (V_{qsn})



Robust end-member modelling - a protocol suggestion



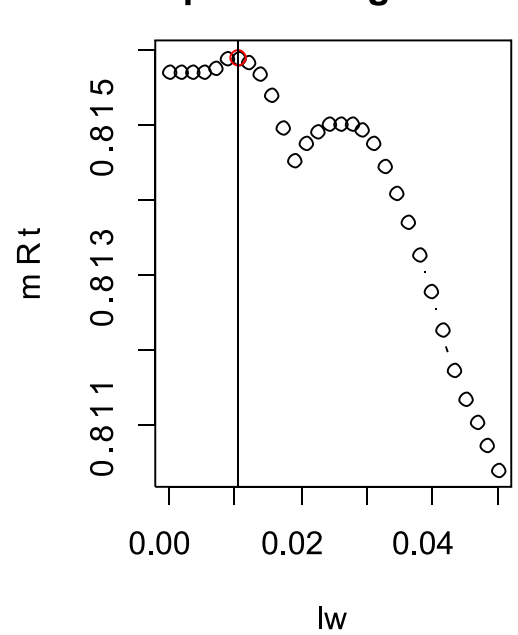
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

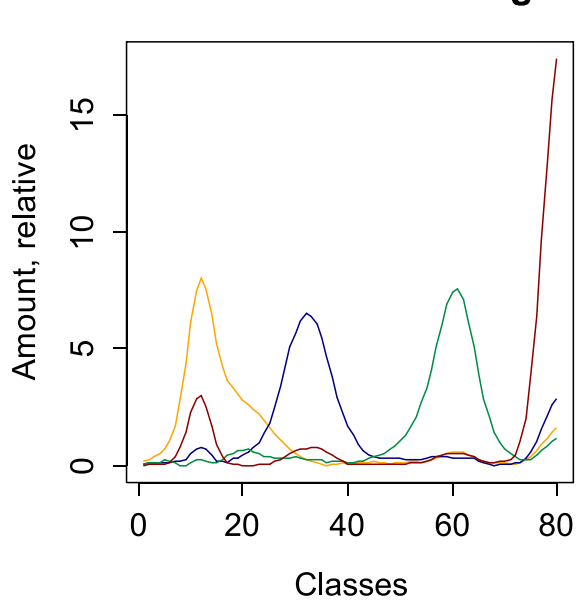
Functions: EMMA()

When all parameters are known and optimised, EMMA() can be run.

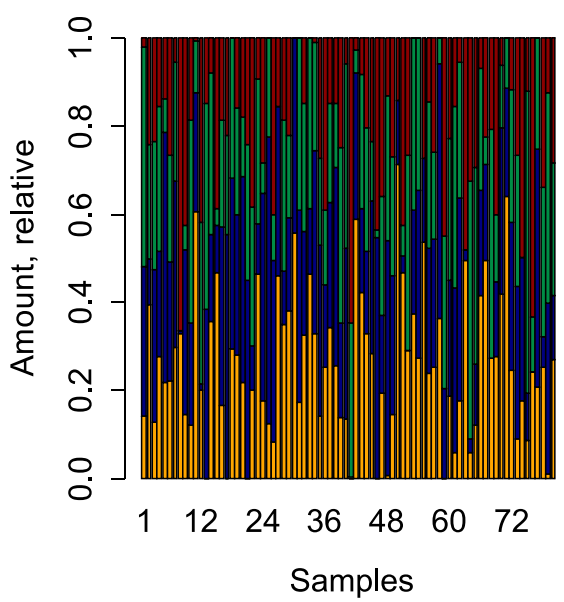
Optimal weight limit



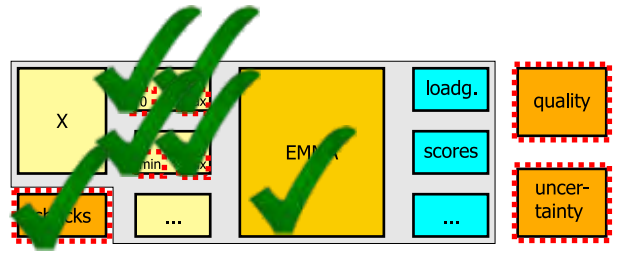
End-member loadings



End-member scores



Robust end-member modelling - a protocol suggestion

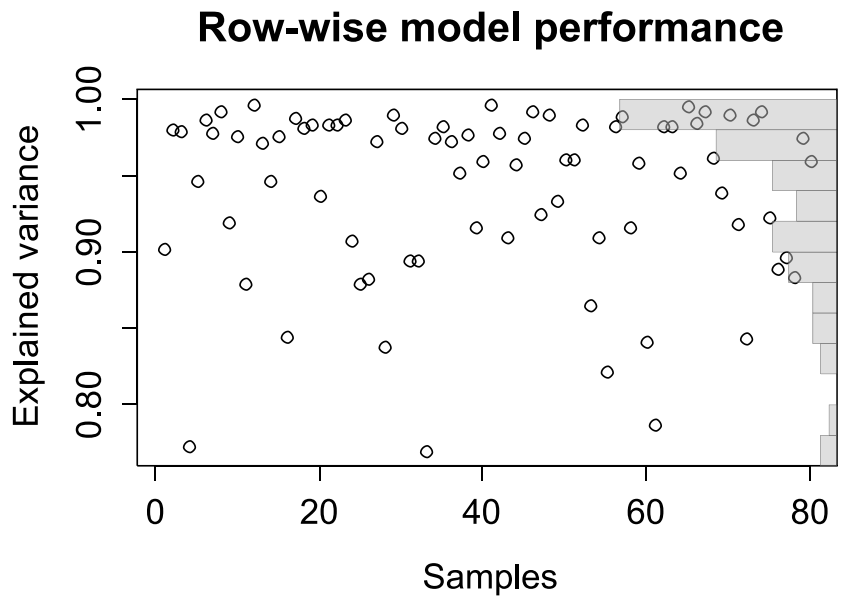


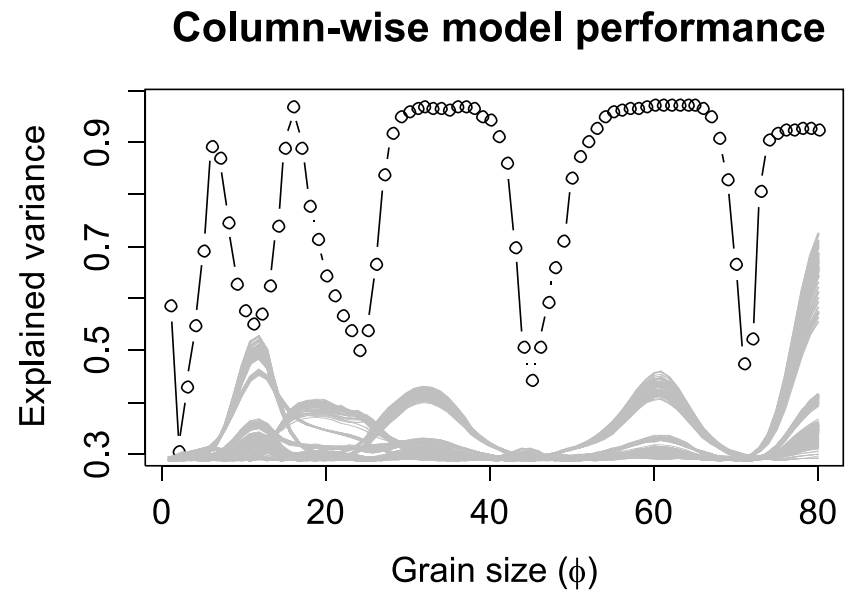
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

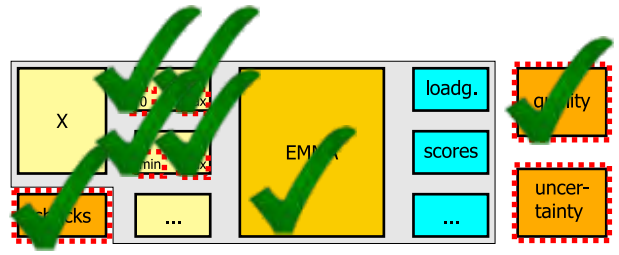
Functions: EMMA()

The quality of the model can be analysed row-wise (sample space) and column-wise (class-wise).





Robust end-member modelling - a protocol suggestion



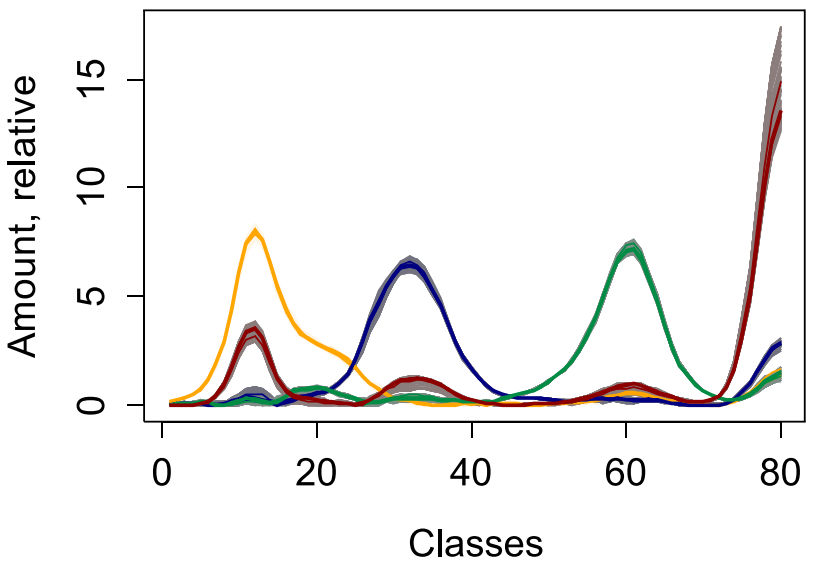
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

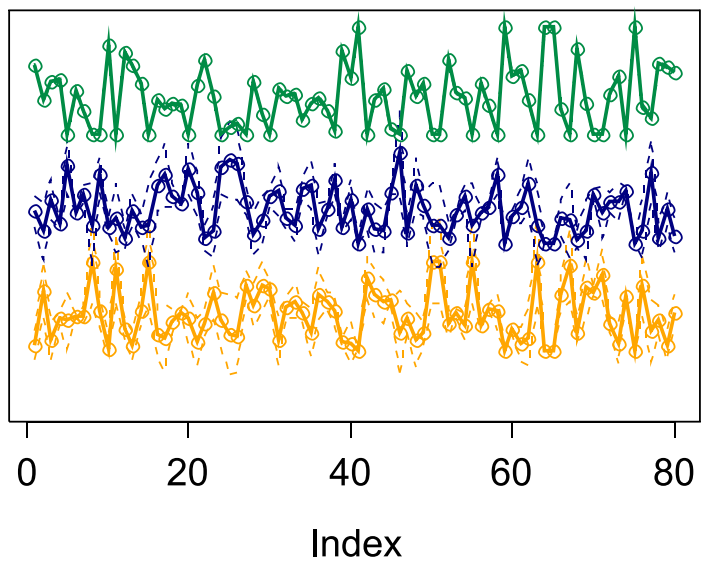
Functions: test.robustness(), Mqs.uncertainty()

The uncertainty of the model can be analysed both, for loadings and scores.

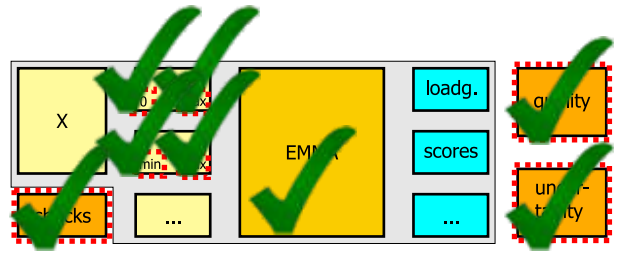
End-member loadings (V_{qsn})



End-member scores with uncertainty



Robust end-member modelling - a protocol suggestion



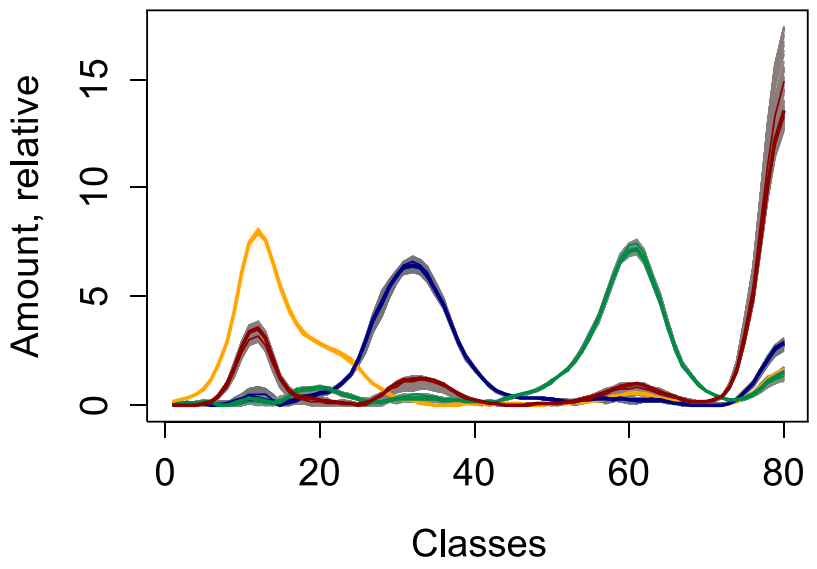
Protocol steps

1. check the data
2. bracket weight transformation limits
3. bracket number of end-members
4. calculate and extract robust end-members
5. run EMMA with optimal parameters
6. evaluate model quality
7. estimate model uncertainty

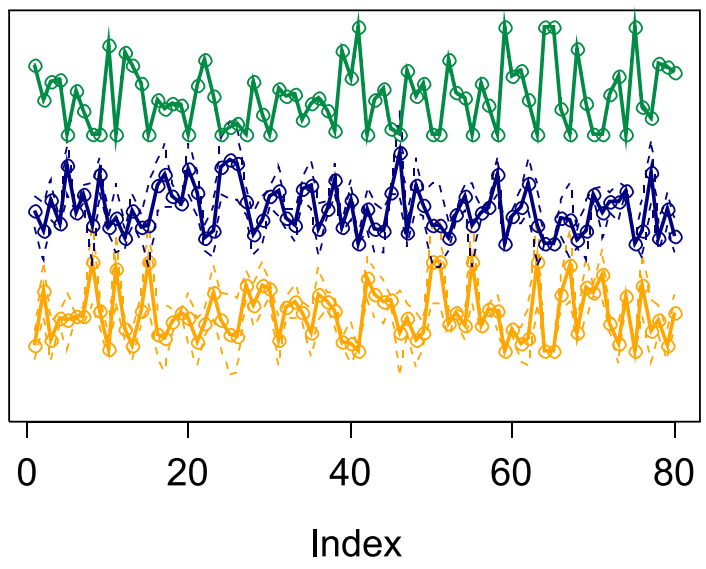
Functions: test.robustness(), Mqs.uncertainty()

The uncertainty of the model can be analysed both, for loadings and scores.

End-member loadings (V_{qsn})



End-member scores with uncertainty



Validation and testing of the algorithm

Pick four process end-members

Measure their individual grain-size distributions

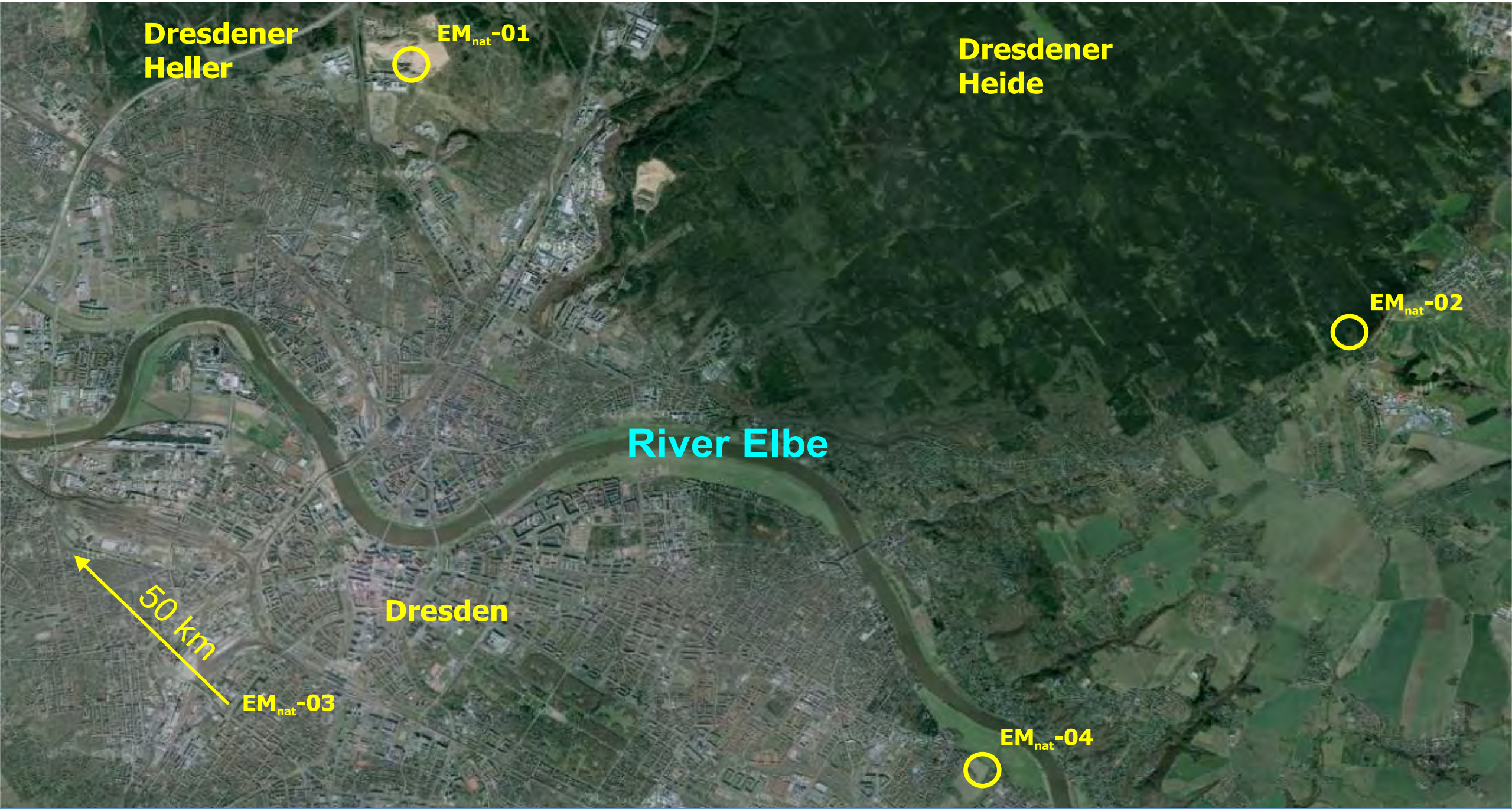
Mix them with given fractions

Measure the mixed samples (n = 100)

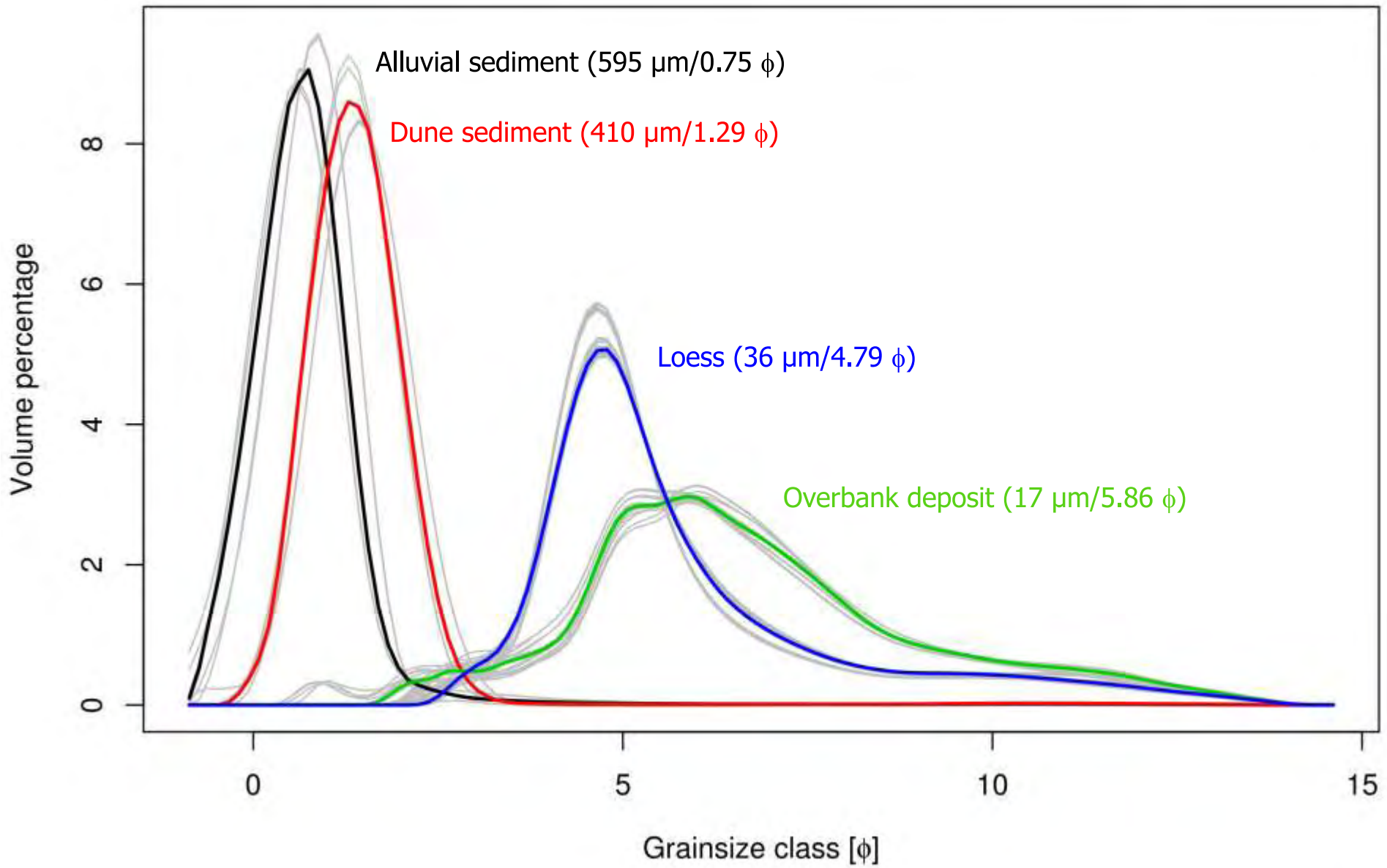
Perform EMMA on mixed samples

Evaluate model outputs

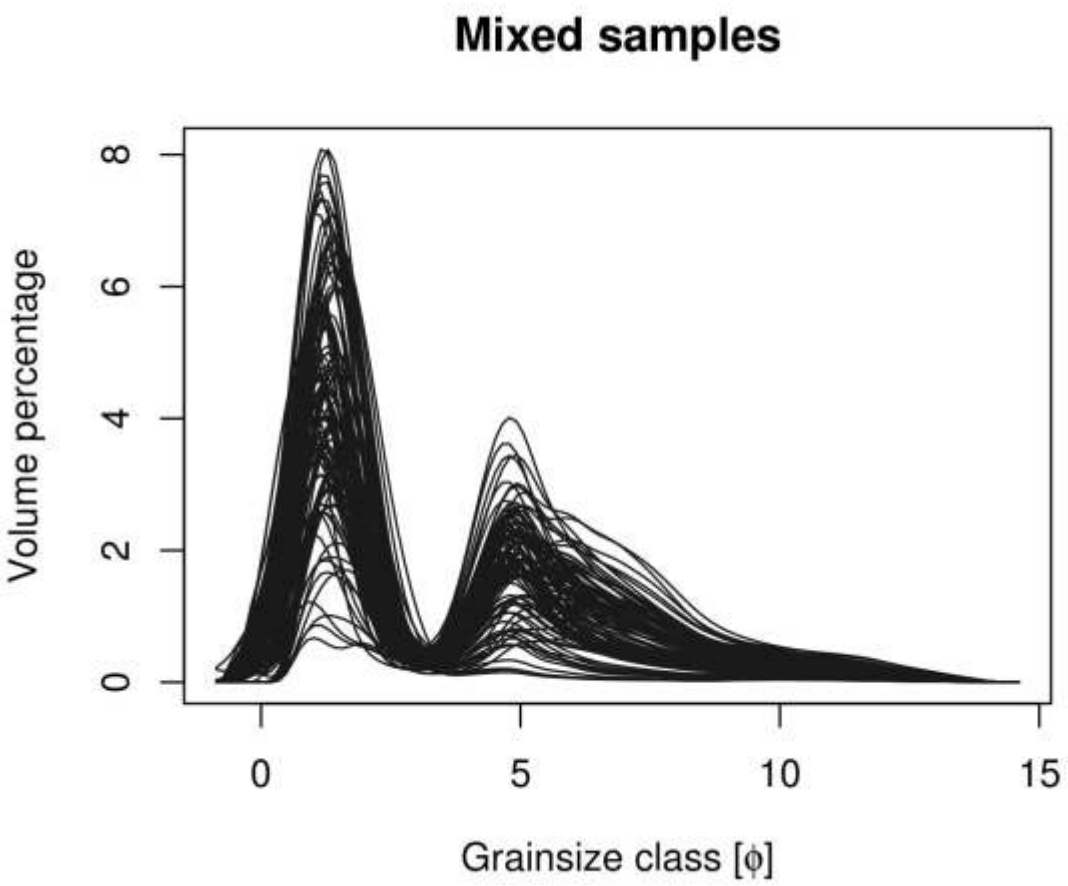
loadings (i.e. end-member shape)
scores (i.e. end-member contribution to each sample)
performance with only three end-members
protocol efficiency
...

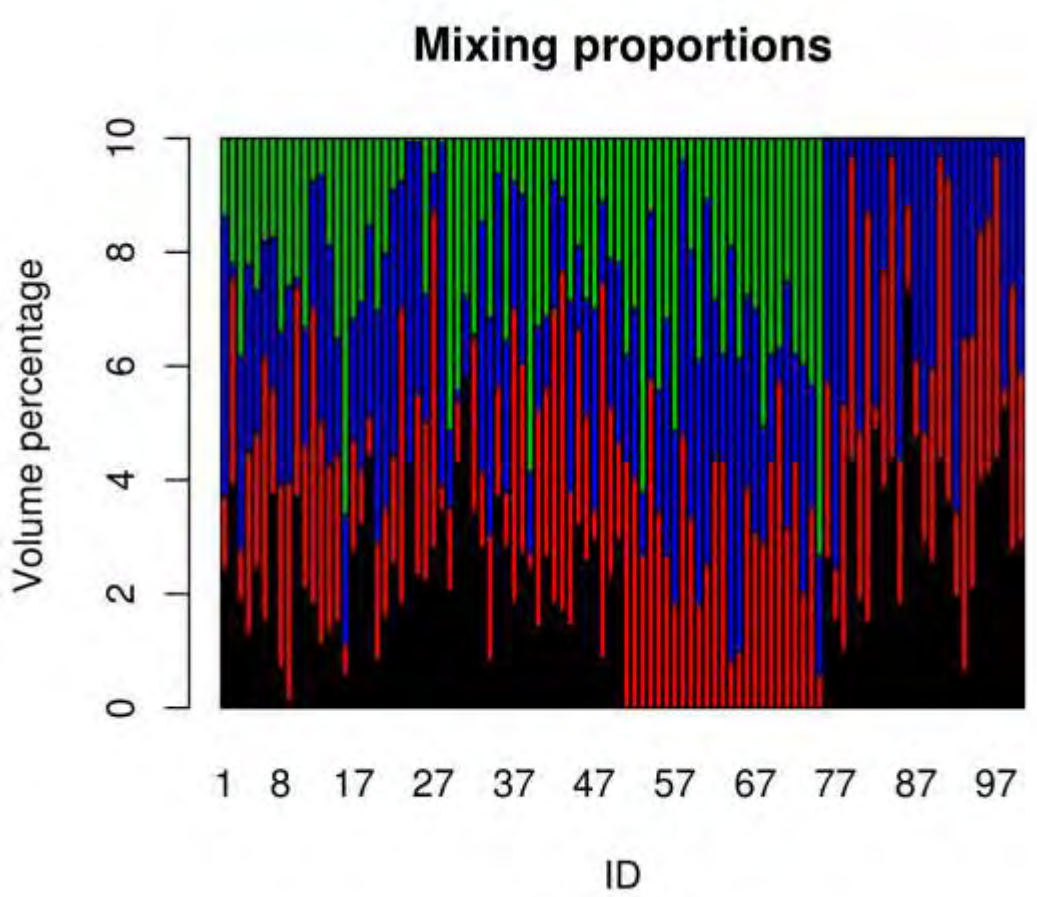

Dresdener Heller
EMnat-01
Dresdener Heide
EMnat-02
River Elbe
50 km
Dresden
EMnat-03
EMnat-04

Grain-size distribution of four natural process end-members

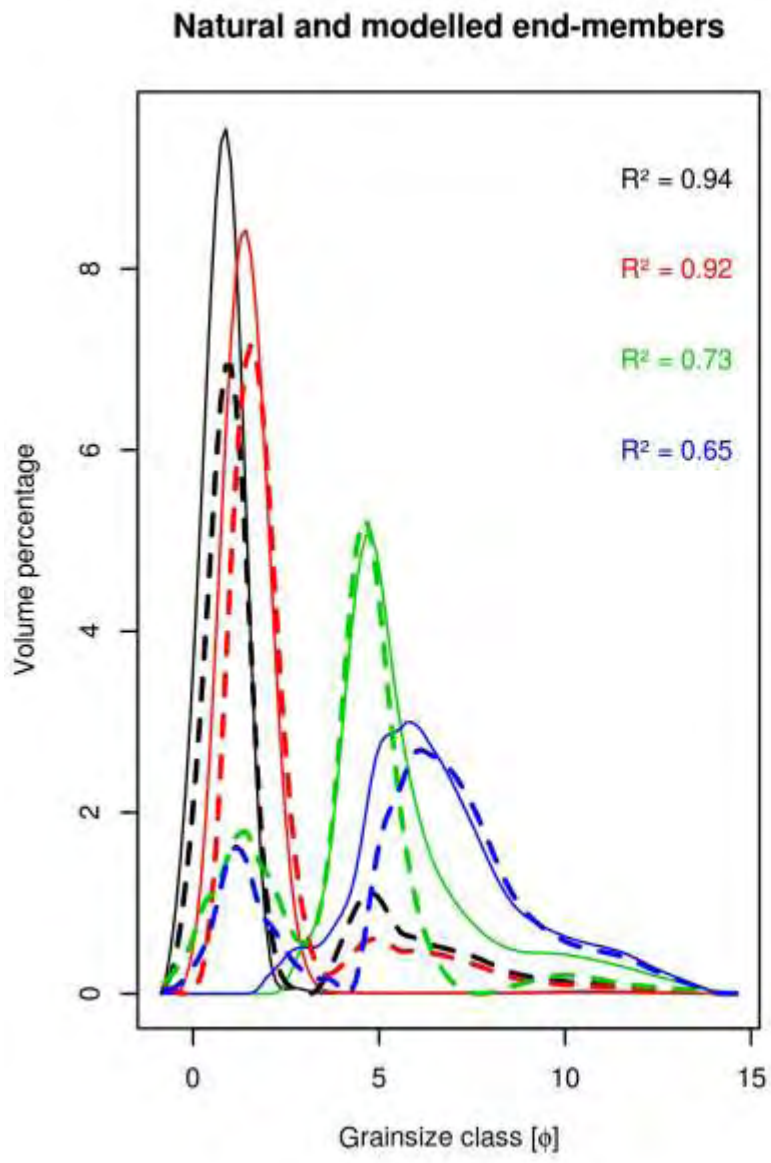


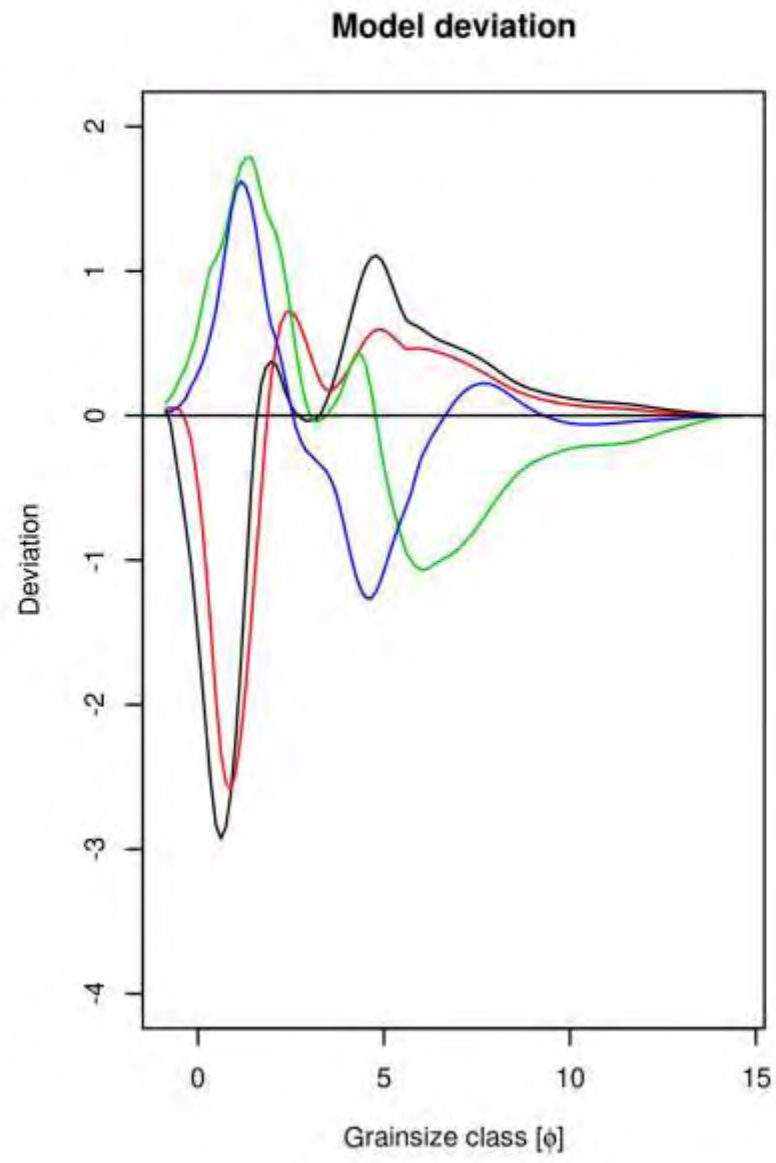
Grain-size distribution and mixing proportions of mixed samples

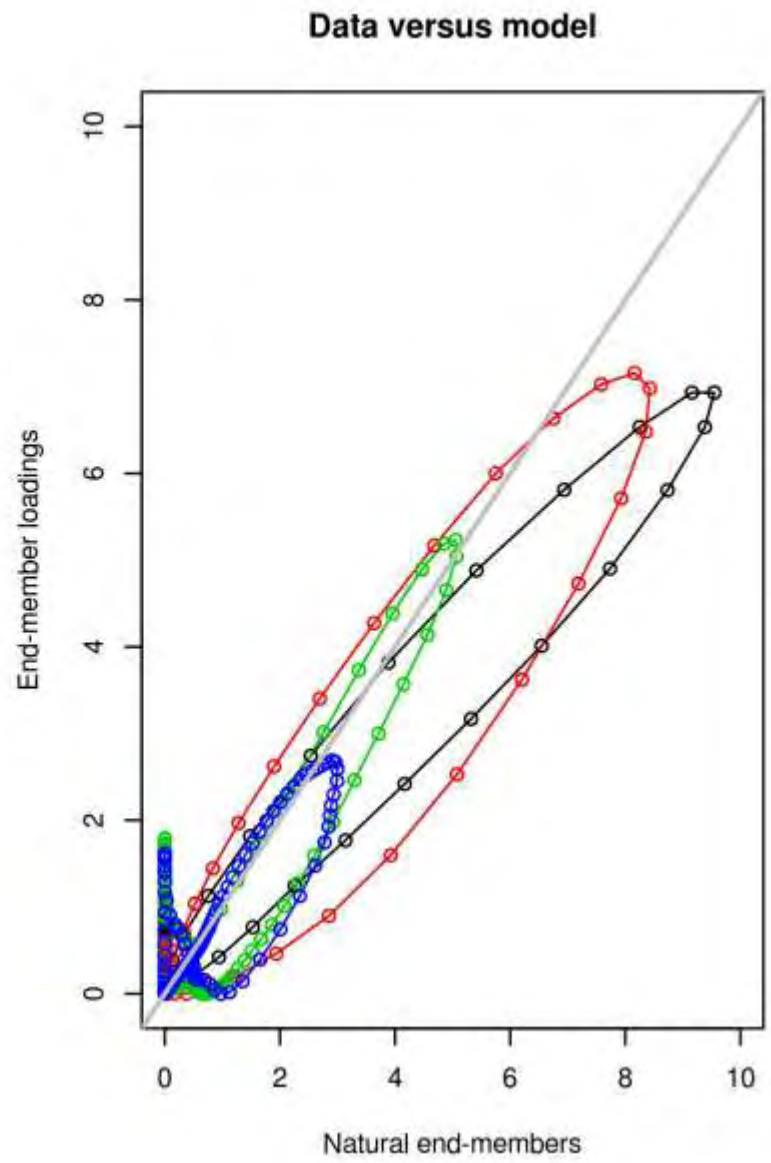




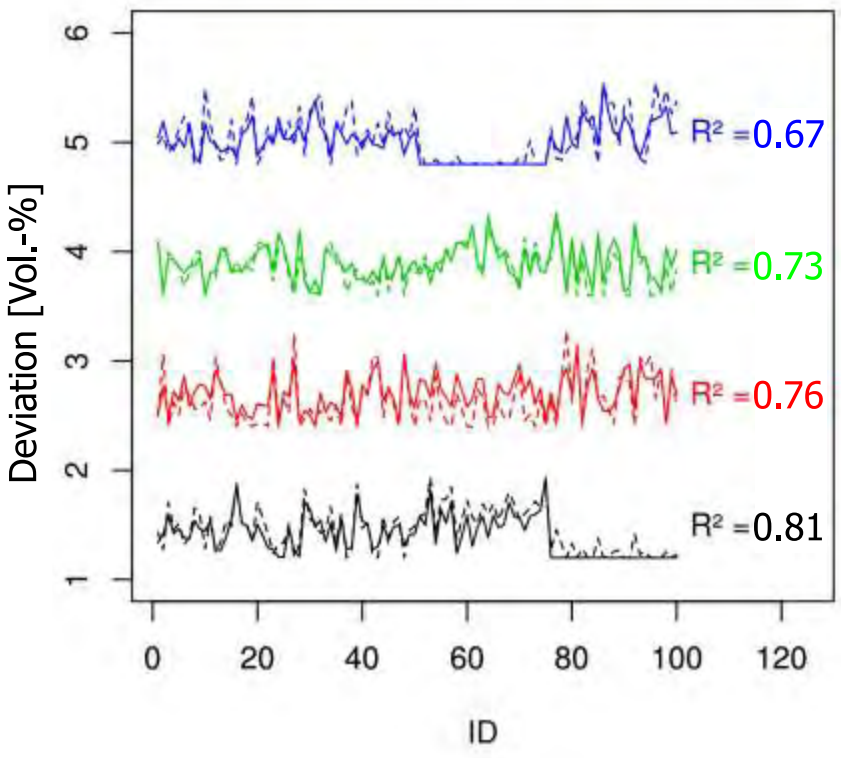
Visual validation of the model - loadings

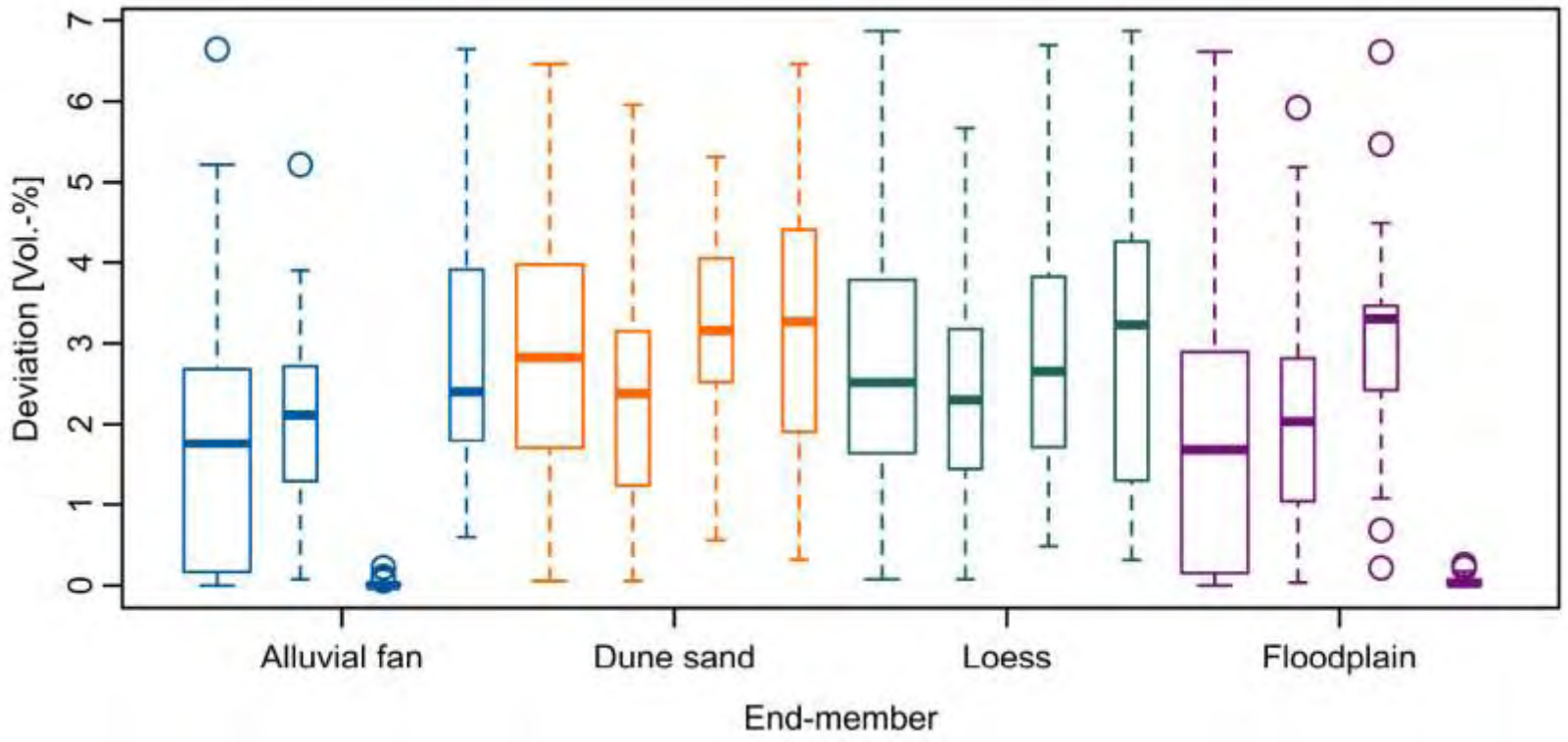




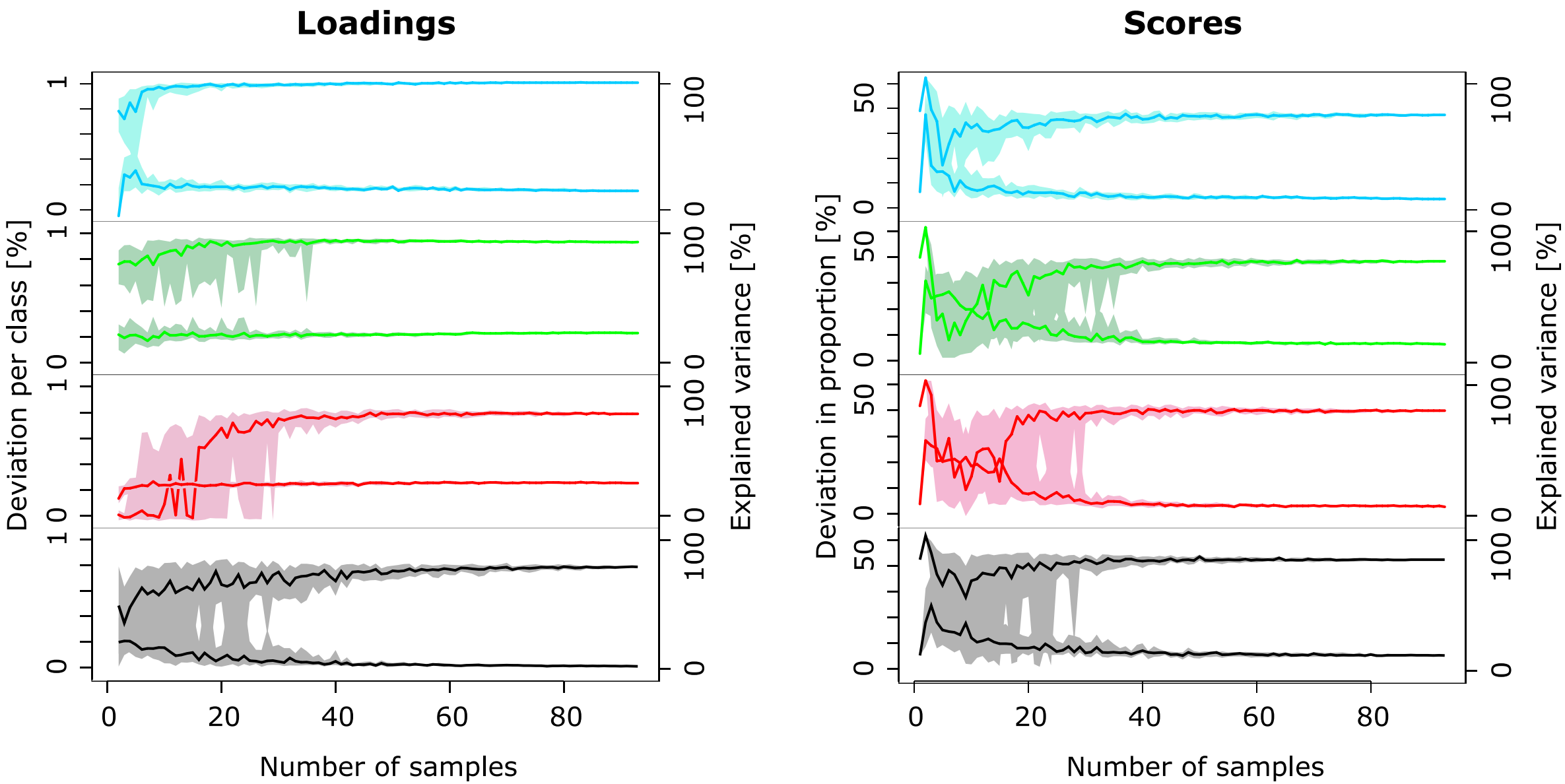


Visual validation of the model - scores



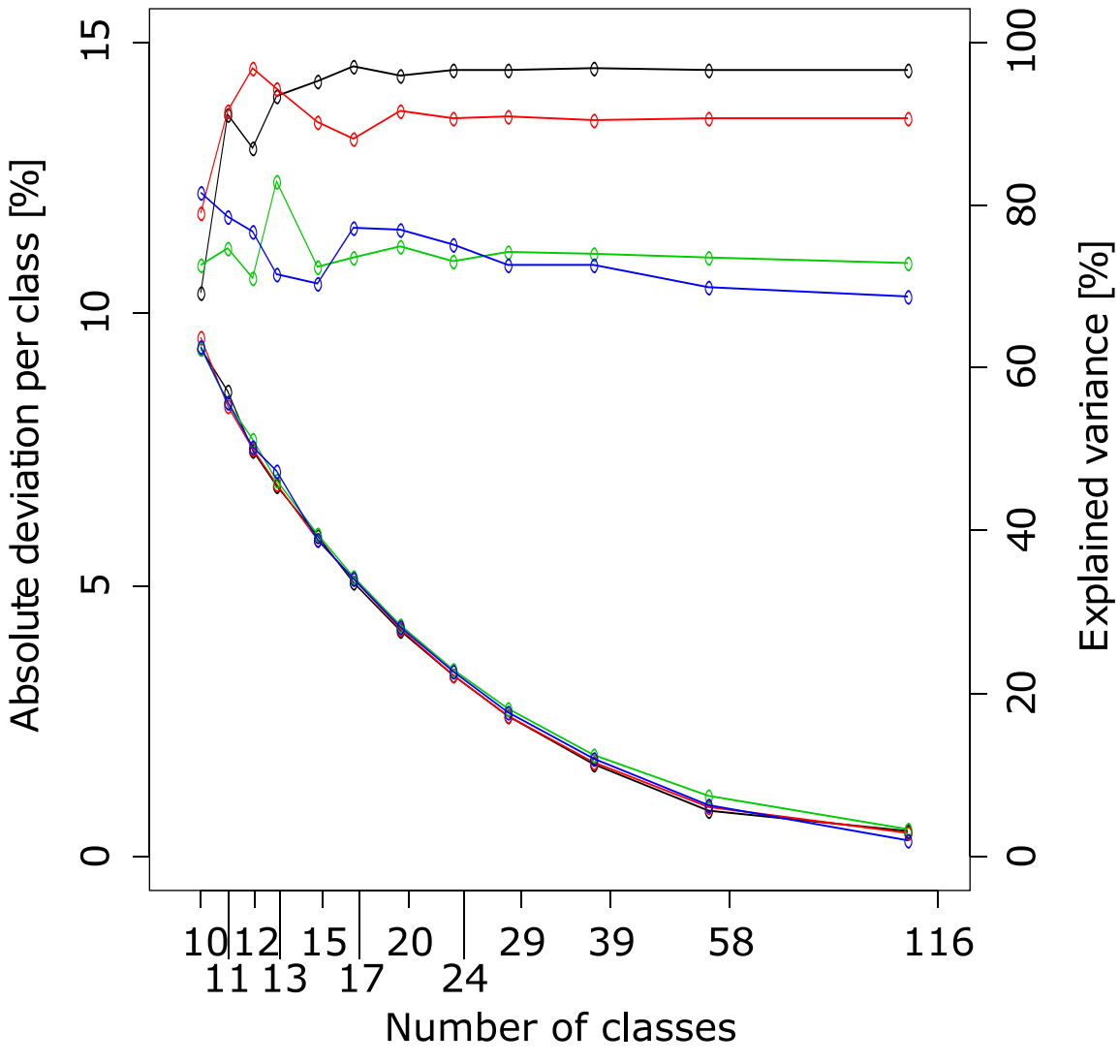


Visual validation of the model - number of samples

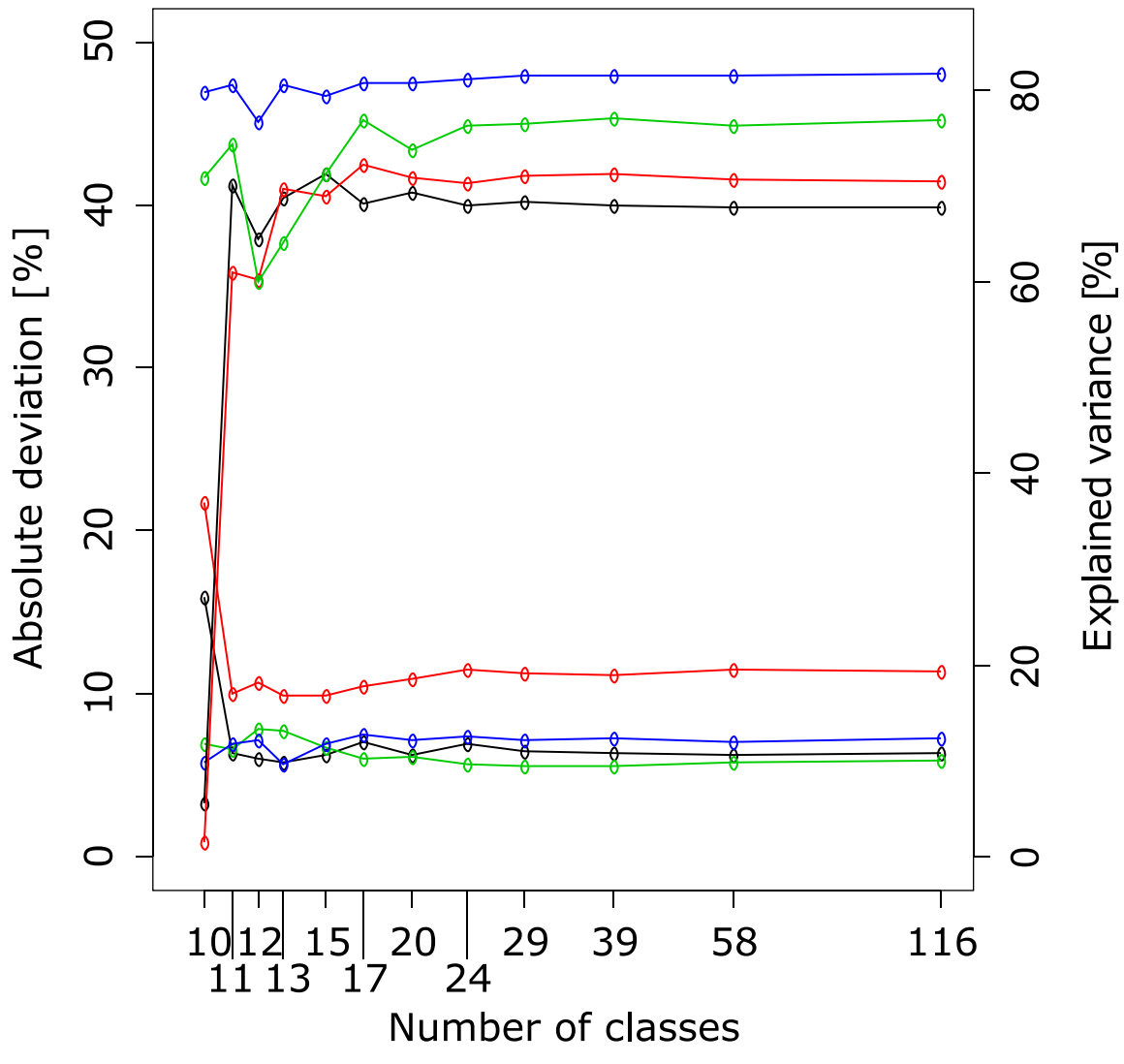


Visual validation of the model - number of classes

Loadings



Scores



Thank you!