Package 'BCE'

February 19, 2015

Version 2.1

Title Bayesian composition estimator: estimating sample (taxonomic) composition from biomarker data

Author Karel Van den Meersche <karel.van_den_meersche@cirad.fr>, Karline Soetaert <karline.soetaert@nioz.nl>

Maintainer Karel Van den Meersche <karel.van_den_meersche@cirad.fr>

Depends R (>= 2.0.1), FME, limSolve, Matrix

Description
Function to estimates taxonomic compositions from biomarker data, using a Bayesian approach.

License GPL (>= 2)

LazyData yes

Repository CRAN

Repository/R-Forge/Project bce

Repository/R-Forge/Revision 65

Repository/R-Forge/DateTimeStamp 2014-05-22 15:06:28

Date/Publication 2014-05-22 19:53:10

NeedsCompilation no

R topics documented:

BCE-package *The Bayesian Compositional Estimator Package*

Description

Functions that estimate sample (taxonomic) composition from biomarker data

`bce1` estimates probability distributions of a sample composition based on an **input ratio matrix**, `A`, containing biomarker ratios in (field) samples, and an **input data matrix**, `B`, containing the biomarker ratios for several taxonomic groups

`tlsce` estimates the total least squares solution

This version differs from the previous version <2 in that a different MCMC method is used that shows better convergence properties and has a more sound underlying statistical model.

Details

Package:	BCE
Type:	Package
License:	GNU Public License 2 or above

Author(s)

Karel Van den Meersche (Maintainer)

Karline Soetaert

References

Van den Meersche, K., K. Soetaert and J.J. Middelburg (2008) *A Bayesian compositional estimator for microbial taxonomy based on biomarkers*, Limnology and Oceanography Methods 6, 190-199

See Also

`bce` the new function, with better convergence properties

`BCE` the original function (versions < 2)

`tlsce` total least squares compositional estimator

Examples

```
## Not run:
## show examples (see respective help pages for details)
example(bce)
example(tlsce)
```

```
## show package vignette
browseURL(paste(system.file(package = "BCE"), "/doc", sep = ""))

## End(Not run)
```

BCE *Bayesian Composition Estimator*

Description

this function is now superseded by the alternative `link{bce1}`**.**

estimates probability distributions of a sample composition based on an **input ratio matrix**, `Rat`, containing biomarker ratios in (field) samples, and an **input data matrix**, `Dat`, containing the biomarker ratios for several taxonomic groups

Usage

```
BCE(Rat, Dat, relsdRat = 0, abssdRat = 0, minRat = 0,
  maxRat = +Inf, relsdDat = 0, abssdDat = 0, tol = 1e-4, tolX = 1e-4,
  positive = 1:ncol(Rat), iter = 100, outputlength = 1000,
  burninlength = 0, jmpRat = 0.01, jmpX = 0.01, unif = FALSE,
  verbose = TRUE, initRat = Rat, initX = NULL, userProb = NULL,
  confInt = 2/3, export = FALSE, file = "BCE")
```

Arguments

`Rat`	initial ratio matrix. Each row of `Rat` contains the biomarker composition of one taxon. As a result of the Bayesian procedure, this initial ratio matrix will be altered.
`Dat`	initial data matrix. Each row of `Dat` contains the biomarker composition of one (field) sample.
`relsdRat`	relative standard deviation on ratio matrix. Either one number or a matrix with the same dimensions as `Rat`.
`abssdRat`	absolute standard deviation on ratio matrix. Either one number or a matrix with the same dimensions as `Rat`.
`minRat`	minimum values of ratio matrix. Either one number or a matrix with the same dimensions as `Rat`.
`maxRat`	maximum values of ratio matrix. Either one number or a matrix with the same dimensions as `Rat`.
`relsdDat`	relative standard deviation on data matrix. Either one number or a matrix with the same dimensions as `Dat`.
`abssdDat`	absolute standard deviation on data matrix. Either one number or a matrix with the same dimensions as `Dat`.
`tol`	minimum standard deviation for data matrix `Dat`. One value.

tolX	minimum x values. Used for MCMC initiation. One value.
positive	A vector containing numbers of columns that should contain strictly positive data. Only these columns are rescaled. The other columns (not in positive) are not rescaled, and can become negative.
iter	number of iterations for MCMC.
outputlength	number of iterations kept in the output.
burninlength	number of initial iterations to be removed from output.
jmpRat	jump length of the ratio matrix Rat (in normal space). Either a number, a vector with length equal to the number of biomarkers (number of columns in Rat), or a or matrix with the same dimensions as the ratio matrix rat.
jmpX	jump length of the composition matrix (in a simplex). Either one number, a vector of length equal to the number of taxa (number of rows in Rat) or a matrix with the same dimensions = c(number of taxa, number of field samples).
unif	logical; if TRUE a uniform distribution for ratio matrix is used. This is similar as in chemtax.
verbose	logical; if TRUE, extra information is provided during the run of the function, such as extra warnings, elapsed time and expected time until the end of the MCMC.
initRat	ratio matrix used to start the markov chain: defaults to the initial ratio matrix.
initX	composition matrix used to start the markov chain: default the LSEI solution of Ax=B.
userProb	function taking two arguments: ratio matrix RAT and composition matrix X, and returning the posterior probability. Dependence of the probability on the data should be incorporated in the function. If not specified, the default probability distribution is the product of a non-informative distribution on the composition matrix, and gamma distributions for the ratio matrix and the data given the model output.
confInt	confidence interval in output; because the distributions may not be symmetrical, standard deviations are not always a useful measure; instead, upper and lower boundaries of the given confidence interval are given. Default is 2/3, i.e there is a probability of 0.66 for a value to be contained within the interval.
export	logical; if TRUE, the function export.bce is called and a list of variables and plots are exported to the specified file.
file	Only if export is TRUE. If not NULL, a character string specifying the file to which objects are saved.

Details

The function BCE searches probability distributions for all elements of a taxonomical composition matrix X and a ratio matrix Rat for which:

$$X\% * \%Rat \simeq Dat$$

It does this by returning iter samples for X and Rat, organized in three-dimensional arrays. The input data matrix Dat and ratio matrix Rat should be in the following formats, with the relative concentrations per biomarker organized in columns:

data matrix:

	marker1	marker2	marker3	marker4
sample1	0.14	0.005	0.35	0.033
sample2	0.15	0.004	0.36	0.034
sample3	0.13	0.004	0.31	0.030
sample4	0.13	0.005	0.33	0.031
sample5	0.14	0.008	0.33	0.036
sample6	0.11	0.082	0.34	0.044

and ratio matrix:

	marker1	marker2	marker3	marker4
species1	0.27	0.13	0.35	0.076
species2	0.084	0	0.5	0.24
species3	0.195	0.3	0	0.1
species4	0.06	0	0	0
species5	0	0	0	0
species6	0	0	0	0

Value

A bce (bayesian compositional estimator) object; a list containing 4 elements

`Rat`	Array with dimension c(nrow(`Rat`),ncol(`Rat`), `iter`) containing the random walk values of the ratio matrix `Rat`.
`X`	Array with dimension c(nrow(`X`),ncol(`X`),`iter`) containing the random walk values of the composition matrix `X`.
`logp`	vector with length `iter` containing the random walk values of the (log) posterior probability.
`naccepted`	integer indicating the number of runs that were accepted.

Note

Producing sensible output:

Markov Chain Monte Carlo simulations are not as straightforward as one might wish; several preliminary runs might be necessary to determine the desired number of iterations, burn-in length and jump length. For all estimated values of Rat and X, their trace (evolution of the values over all iterations) has to display random behaviour; no obvious trends should appear. A few parameters can be tuned to obtain such behaviour:

- **jump length** The jump length determines how big the jumps are for each step in the random walk. A longer jump length will make you jump around faster in the parameter space, but acceptance of new points can get very low. Smaller jump lengths increase the acceptance rate, but the algorithm will move too slowly, and a lot more runs will be needed to scan the whole parameter space. A good way to find a good jump length, is look at the number of points accepted. If the output is saved under the name `MCMC`, you can find the number of accepted points under `MCMC$naccepted`. It is also given if you run the model with `verbose=TRUE` (default). This value should be somewhere between 5% and 40%. For long runs, 5 % can be acceptable, for short runs, you will prefer a higher acceptance in order to have enough different points.

20% accepted is usually a good number. Do some preliminary runs with `iter=1000-10000` and tune the jump length parameters `jmpRat` and `jmpX`. You can set different jump lengths for each column of the ratio matrix, or 1 jump length for the whole ratio matrix, and 1 jump length for the composition matrix. Decreasing the jump lengths will generally increase the acceptance rate and vice versa. Also the mixing rate (the speed with which accepted points change their values) will be influenced. You want this mixing rate to be as high as possible, whilst maintaining enough accepted points.

- **burninlength** The program uses the solution of lsei using the original ratio matrix as starting values for the MCMC. This might in some cases be far from the optimal solution, and the MCMC algorithm will start with moving towards this optimal solution. This is called a burn-in. When there is a slow mixing rate, this can take a considerable number of cycles. As it can influence the averages and standard deviations, you might want to remove it from the mcmc objects. By defining a burnin length, the first '`burninlength`' cycles will not be written to the output. Look at some plots to determine if you need to specify a burnin length.
- **iter** the number of iterations: start with 10000 runs or less; check the output and estimate how many runs you will need to get a random pattern in the output.

Author(s)

Karel Van den Meersche <k.vdmeersche@nioo.knaw.nl>, Karline Soetaert <k.soetaert@nioo.knaw.nl>.

References

Van den Meersche, K., K. Soetaert and J.J. Middelburg (2008) *A Bayesian compositional estimator for microbial taxonomy based on biomarkers*, Limnology and Oceanography Methods 6, 190-199

See Also

`summary.bce`, `plot.bce`, `export.bce`, `pairs.bce`

Examples

```
##====================================

# example using bceInput data
# first try

X <- BCE(bceInput$Rat,bceInput$Dat,relsdRat=.2,relsdDat=.2,
         iter=1000,outputlength=5000,jmpX=.01,jmpRat=.01)

## the number of accepted runs is too low;
## we play around with the jump lengths jmpX and jmpRat

X <- BCE(bceInput$Rat,bceInput$Dat,relsdRat=.2,relsdDat=.2,
         iter=1000,outputlength=5000,jmpX=.02,jmpRat=.002)

## we inspect the output:
plot(X)

## For every element of X and Rat, we want to obtain a well-mixed,
```

```
## random trace. In this case, mixing is still a little poor.
## to optimize mixing in the ratio matrix, it is a good idea
## to make the jump length linear to the ratio matrix
## standard deviation (sdrat=.2*rat) :
X <- BCE(bceInput$Rat,bceInput$Dat,relsdRat=.2,relsdDat=.2,
         iter=1000,outputlength=5000,jmpX=.02,
         jmpRat=.2*(.2*bceInput$Rat))
plot(X)

## mixing improved a lot; we repeat the run with more iterations
## to improve the reliability of the results.
## the following run can take a few minutes - so it is toggled off
#X <- BCE(bceInput$Rat,bceInput$Dat,relsdRat=.2,relsdDat=.2,
#         iter=100000,outputlength=5000,jmpX=.02,
#         jmpRat=.2*(.2*bceInput$Rat))
#plot(X)
## you can see in the plots that traces for all elements of Rat and X
## are well-mixed. This run was saved in "bceOutput"

Sum <-summary(bceOutput)

# show results as mean with ranges
print(Sum$meanX)

# plot estimated means and ranges (lbX=lower, ubX=upper bound)
xlim <- range(c(Sum$lbX,Sum$ubX))

# first the mean
dotchart(x=t(Sum$meanX),xlim=xlim,
         main="Taxonomic composition",
         sub="using bce",pch=16)

# then ranges
nr <- nrow(Sum$meanX)
nc <- ncol(Sum$meanX)

for (i in 1:nr)
{ip <-(nr-i)*(nc+2)+1
 cc <- ip : (ip+nc-1)
 segments(t(Sum$lbX[i,]),cc,t(Sum$ubX[i,]),cc)
 }

# show results as pairs plot
pairs(bceOutput,sample=3,main="Station 3")
```

Description

This function estimates taxonomic compositions of algal communities based on biomarker field data. More precisely, it estimates the probability distributions of a sample composition based on an **input ratio matrix**, A that contains prior estimates of biomarker ratios in different taxa, and an **input data matrix**, B, containing biomarker ratios measured in field samples.

Probability distributions are estimated based on an adaptive metropolis MCMC method, function modMCMC from package FME.

Usage

```
bce1(A, B, Wa=NULL, Wb=NULL,
       jmpType="default", jmpA=.1,jmpX=.1, jmpCovar=NULL,
       initX=NULL, initA=NULL, priorA="normal", minA=NULL, maxA=NULL,
       var0=NULL, wvar0=1e-6, Xratios=TRUE, verbose=TRUE, ...)
```

Arguments

A	input (group) ratio matrix; can be a matrix or a dataframe
B	input (field) data matrix; can be a matrix or a dataframe
Wa	elementwise weight matrix for A, with the same dimensions as A. $$[Wa * (A - A_0)]^2$$ is minimized
Wb	elementwise weight matrix for B, with the same dimensions as B. $$[Wb * (A\% * \%X - B)]^2$$ is minimized
jmpType	one of "default", "estimate" or "covar"; if default, jmpA and jmpX are the jump lengths. if jmpA or jmpX is a number, then this is the jump length for all elements of A resp. X. If "estimate", the initial jump length is proportional to an estimated covariance matrix for the tlsce fit for A and the lsei fit of X (or Q if Xratios). jmpA and jmpX are then used as rescaling factors for the jump covariance matrix. If "covar", a jump covariance matrix with the correct dimensions, obtained from a previous run, is given as parameter jmpCovar. Covariances can be calculated from the result.
jmpA	jump length of A: a number or a matrix with dim(A); see details jmpType
jmpX	jump lenth of X: a number or a matrix with dim(X); see details jmpType
jmpCovar	only if jmpType="covar", the covariance matrix to initiate the jumps - see details jmpType
initX	composition matrix used to start the markov chain: default the tlsce solution of Ax=B
initA	ratio matrix used to start the markov chain: default the input ratio matrix A
priorA	"normal" (gaussian - default) or "uniform".
minA	minimum values for A

maxA	maximum values for A
var0	initial model variance; if 'NULL', then the model variance of tlsce(A,B,...) is used
wvar0	relative weight of the initial model variance (see modMCMC). Ideally this would be 0 (initial model variance is not taken into account); because wvar0=0 is a special case in modMCMC() (fixed model variance), the default value is set to a small number (wvar0=1e-6)
Xratios	does the composition matrix contain ratios (TRUE) or estimated biomass concentrations (TRUE) per sample? In the latter case, B must contain the pigment concentrations as measured in the samples (not rescaled)
verbose	when TRUE will give more verbose output
...	arguments to pass on to modMCMC()

Details

The function bce1 searches probability distributions for all elements of a taxonomical composition matrix X and a ratio matrix A for which:

$$A\% * \%X \simeq B$$

It does this by returning niter samples for A and X, organized in three-dimensional arrays. The input data matrix B and ratio matrix A should be in the following formats, with the relative concentrations per biomarker organized in columns:

data matrix B:

	sample1	sample2	sample3	sample4
marker1	0.14	0.005	0.35	0.033
marker2	0.15	0.004	0.36	0.034
marker3	0.13	0.004	0.31	0.030
marker4	0.13	0.005	0.33	0.031
marker5	0.14	0.008	0.33	0.036
marker6	0.11	0.082	0.34	0.044

and ratio matrix A:

	species1	species2	species3	species4
marker1	0.27	0.13	0.35	0.076
marker2	0.084	0	0.5	0.24
marker3	0.195	0.3	0	0.1
marker4	0.06	0	0	0
marker5	0	0	0	0
marker6	0	0	0	0

Value

An object of class bce and _modMCMC_ (returned by the function modMCMC). This object has methods for the generic functions 'summary', 'plot', 'pairs'- see ?modMCMC. It is distinguished from other modMCMC objects by 3 extra attributes that allow to extract matrices A and X from the mcmc result: "dim_A" (dimensions of A), "A_not_null" (which elements of A are not zero and thus included in the mcmc) and Xratios (whether X was rescaled, yes or no).

Note

Producing sensible output:

Markov Chain Monte Carlo simulations are not as straightforward as one might wish; several preliminary runs might be necessary to determine the desired number of iterations, burn-in length and jump length. For all estimated values of Rat and X, their trace (evolution of the values over all iterations) has to display random behaviour; no obvious trends should appear. A few parameters can be tuned to obtain such behaviour:

- **jump length** The jump length determines how big the jumps are for each step in the random walk. A longer jump length will make you jump around faster in the parameter space, but acceptance of new points can get very low. Smaller jump lengths increase the acceptance rate, but the algorithm will move too slowly, and a lot more runs will be needed to scan the whole parameter space. A good way to find a good jump length, is look at the number of points accepted. If the output is saved under the name `MCMC`, you can find the number of accepted points under `MCMC$naccepted`. It is also given if you run the model with `verbose=TRUE` (default). This value should be somewhere between 5% and 40%. For long runs, 5 % can be acceptable, for short runs, you will prefer a higher acceptance in order to have enough different points. 20% accepted is usually a good number. Do some preliminary runs with `niter=1000-10000` and tune the jump length parameters `jmpRat` and `jmpX`. You can set different jump lengths for each column of the ratio matrix, or 1 jump length for the whole ratio matrix, and 1 jump length for the composition matrix. Decreasing the jump lengths will generally increase the acceptance rate and vice versa. Also the mixing rate (the speed with which accepted points change their values) will be influenced. You want this mixing rate to be as high as possible, whilst maintaining enough accepted points.
- **burninlength** The program uses the solution of lsei using the original ratio matrix as starting values for the MCMC. This might in some cases be far from the optimal solution, and the MCMC algorithm will start with moving towards this optimal solution. This is called a burn-in. When there is a slow mixing rate, this can take a considerable number of cycles. As it can influence the averages and standard deviations, you might want to remove it from the mcmc objects. By defining a burnin length, the first '`burninlength`' cycles will not be written to the output. Look at some plots to determine if you need to specify a burnin length.
- **niter** the number of iterations: start with 10000 runs or less; check the output and estimate how many runs you will need to get a random pattern in the output.

Author(s)

Karel Van den Meersche <k.vdmeersche@nioo.knaw.nl>, Karline Soetaert <k.soetaert@nioo.knaw.nl>.

References

Van den Meersche, K., K. Soetaert and J.J. Middelburg (2008) *A Bayesian compositional estimator for microbial taxonomy based on biomarkers*, Limnology and Oceanography Methods 6, 190-199

See Also

summary.bce, plot.bce, export.bce, pairs.bce

Examples

```
##====================================

# example using bceInput data
# !!! should be weighted to correspond better to example of BCE!!!
A <- t(bceInput$Rat)
B <- t(bceInput$Dat)


result <- bce1(A,B,niter=1000)

## the number of accepted runs is zero;
## try different starting values

result <- bce1(A,B,niter=1000,initX=matrix(1/ncol(A),ncol(A),ncol(B)))

## number of accepted runs is still low;
## smaller jumps

result <- bce1(A,B,niter=1000,initX=matrix(1/ncol(A),ncol(A),ncol(B)),jmpA=.01,jmpX=.01)
Sum <-summary(result)

## did the algorithm converge?
plot(result$SS,type="l")
## no


## more runs, using the output of previous run as input.
result <- bce1(A,B,niter=1e4,jmpA=.01,jmpX=.01,updatecov=1e3,
               initX=Sum$lastX,initA=Sum$lastA,
               jmpCovar=Sum$covar*2.4^2/ncol(result$pars),
               )
Sum <-summary(result)

## we inspect the output:
plot(result$SS,type="l")
plot(result,ask=TRUE)
## looks already pretty good; to get a better result, repeat one more
## time with a  longer run. Uncomment the following paragraph and run.
## go get some coffee, this might take a while (~30s).

## result <- bce1(A,B,niter=1e5,jmpA=.01,jmpX=.01,updatecov=1e3,
##                outputlength=1e3,burninlength=.35e5,
```

```
##                initX=Sum$lastX,initA=Sum$lastA,
##                jmpCovar=Sum$covar*2.4^2/ncol(result$pars),
##                )
## Sum <-summary(result)
## plot(result$SS,type="l")
## plot(result,ask=TRUE)

# show results as mean with ranges
print(Sum$meanX)

# plot estimated means and ranges (lbX=lower, ubX=upper bound)
xlim <- range(c(Sum$lbX,Sum$ubX))

# first the mean
dotchart(x=t(Sum$meanX),xlim=xlim,
         main="Taxonomic composition",
         sub="using bce",pch=16)

# then ranges
nr <- nrow(Sum$meanX)
nc <- ncol(Sum$meanX)

for (i in 1:nr)
  {ip <-(nr-i)*(nc+2)+1
   cc <- ip : (ip+nc-1)
   segments(t(Sum$lbX[i,]),cc,t(Sum$ubX[i,]),cc)
  }
```

---

bceInput *ratiomatrix and datamatrix for demonstration of BCE().*

Description

The datamatrix contains a set of biomarker measurements for a number of field samples.

The ratiomatrix contains biomarker data of a number of biological taxa. `BCE()` uses these matrices to estimate the taxonomical composition of the samples based on the provided taxa. For use with the function `bce1()`, they have to be transposed.

Usage

```
bceInput
```

Examples

```
##====================================
## Graphical representation of the example input data
palette(rainbow(12, s = 0.6, v = 0.75))
```

```
mp     <- apply(bceInput$Rat,MARGIN=2,max)
mp2    <- apply(bceInput$Dat,MARGIN=2,max)
pstars <- rbind(t(t(bceInput$Rat)/mp),t(t(bceInput$Dat)/mp2))

stars(pstars, len = 0.9, key.loc = c(7.2, -2),scale=FALSE,
      ncol=5,ylim=c(0,3),main = "bce Input: species + field samples",
      draw.segments = TRUE, flip.labels=FALSE)
```

---

bceOutput *bce output generated by running the bceInput example*

Description

Result generated by running BCE using data `bceInput` as input.

the run was initiated with the following command:

```
bceOutput <- BCE(bceInput$Rat,bceInput$Dat,relsdRat=.2,relsdDat=.2,
iter=100000,outputlength=5000,jmpX=.02,jmpRat=.2*(.2*bceInput$Rat))
```

this run took several minutes.

Usage

```
bceOutput
```

Examples

```
summary(bceOutput)$meanX
```

---

export.bce *export BCE*

Description

export function: writes a BCE-object and its summary statistics to a series of files.

Usage

```
export.bce(x, file="BCE", input.list=NULL, ...)
```

Arguments

`x`	a bce object, output of the function BCE().
`file`	file to which the bce object is written.
`input.list`	a list of the arguments in bce() can be provided and saved as well.
`...`	additional arguments.

Details

The bce object is saved (save) to the specified file. For people not familiar to R, it can be more 'user-friendly' to export summary results to comma delimited textfiles, that can be easily imported into a spreadsheet program. The function summary.bce is called to calculate summary statistics of a BCE object; These are then written to a series of .csv files with a name that combines the specified filename and a string indicating the content of the .csv files. Traces and marginal probabilities of all estimated values are plotted and saved in .png files. These traces should be inspected carefully before accepting any results (see also plot.bce).

Author(s)

Karel Van den Meersche

See Also

BCE, summary.bce, plot.bce, pairs.bce

Examples

```
## Not run: export.bce(bceOutput, file="bceOutput")
```

---

pairs.bce *Pairs plot of a BCE*

Description

produces pairs plots of the random walks of the BCE.

Usage

```
## S3 method for class 'bce'
 pairs(x, sample = 1, gap = 0,upper.panel = NA,
  diag.panel = NA, ...)
```

Arguments

x	either a bce object or the random walk values (X) of the composition matrix.
sample	the sample number for which the pairs plot is to be drawn.
gap	Distance between subplots, in margin lines - a pairs parameter.
upper.panel	panel function to be used above the diagonal - the default writes the correlations.
diag.panel	panel function to be used on the diagonal - the default plots a histogram.
...	any other parameters passed to function pairs.

Author(s)

Karline Soetaert

See Also

BCE, summary.bce, plot.bce, export.bce

Examples

```
# bceOutput is an example output based on bceInput

pairs(bceOutput,sample=2,main="Station 2")
```

---

plot.bce *plot BCE*

Description

produces summary plots of the random walks of the BCE; these are intended for inspection only.

Usage

```
## S3 method for class 'bce'
plot(x, ...)
```

Arguments

x	an object of class bce and/or _modMCMC_. Output of one of the functions BCE or bce1.
...	additional arguments.

Details

When the argument an object is of class _modMCMC_, which is the case if it is the output of bce1(), it is treated as such (see modMCMC).

Calling the plot-function with the output of BCE as argument, will produce a series of plots with the random walks of each variable. The layout of these plots is kept very sober, as they are primarily intended for inspection of the random walk (see BCE). Users are free to write their own publication quality plots. Click or hit Enter to see the next plot, hit Esc to stop seeing new plots.

Author(s)

Karel Van den Meersche

See Also

BCE, summary.bce, export.bce, pairs.bce

Examples

```
# bceOutput is an example output based on bceInput

plot(bceOutput)
```

`rescaleRows` *rescale rows*

Description

returns a row-rescaled matrix (`rowSums(.)==1`).

Usage

```
rescaleRows(A, columns=1:ncol(A))
```

Arguments

`A`	matrix or dataframe to be row-rescaled: `rowSums(rescaleRows(A))==1`.
`columns`	vector containing indices of the columns that should be included in the normalisation.

Value

`A`	row-rescaled matrix or partially row-rescaled matrix.

Author(s)

Karel Van den Meersche

`summary.bce` *summary BCE*

Description

basic statistics of a bce object

Usage

```
## S3 method for class 'bce'
summary(object, confInt = 2/3, ...)
```

Arguments

`object`	a bce-object, output of the function bce1() or BCE().
`confInt`	confidence interval of values of composion matrix and ratio matrix.
`...`	additional arguments affecting the summary produced.

Value

if object is output of the function bce1():

a list containing:

meanA	Average solution of the ratio matrix,found through MCMC.
meanX	Average solution of the composition matrix, found through MCMC.
bestA	Ratio matrix for which the posterior probability is maximal.
bestX	Composition matrix for which the posterior probability is maximal.
sdA	standard deviation of the ratio matrix.
sdX	standard deviation of the composition matrix.
lastA	Last value for the ratio matrix in the MCMC run.
lastX	Last value for the composition matrix in the MCMC run.
medianA	Median of the ratio matrix.
medianX	Median of the composition matrix.
ubA	Upper boundary of the confidence interval of the elements of the ratio matrix.
ubX	Upper boundary of the confidence interval of the elements of the composition matrix.
lbA	Lower boundary of the confidence interval of the elements of the ratio matrix.
lbX	Lower boundary of the confidence interval of the elements of the composition matrix.
covar	Covariance matrix of all elements of A and X.

if object is output of the function BCE():

a list containing:

firstX	X determined through least squares regression from the initial ratio matrix and the data matrix.
bestRat	Ratio matrix for which the posterior probability is maximal.
bestX	Composition matrix for which the posterior probability is maximal.
bestp	Maximal posterior probability.
bestDat	Product of bestRat and bestX.
meanRat	Means of the elements of the ratio matrix.
sdRat	Standard deviation of the elements of the ratio matrix.
lbRat	Lower boundary of the confidence interval of the elements of the ratio matrix.
ubRat	Upper boundary of the confidence interval of the elements of the ratio matrix.
covRat	Covariance matrix of the elements of the ratio matrix.
meanX	Means of the elements of the composition matrix.
sdX	Standard deviation of the elements of the composition matrix.
lbX	Lower boundary of the confidence interval of the elements of the composition matrix.
ubX	Upper boundary of the confidence interval of the elements of the composition matrix.
covX	Covariance matrix of the elements of the composition matrix.

Author(s)

Karel Van den Meersche

See Also

bce1, BCE, plot.bce, export.bce, pairs.bce

Examples

```
# bceOutput is an example output based on bceInput

summary(bceOutput)
```

`tlsce` *Total Least Squares Composition Estimator*

Description

estimates a matrix X for which:

$$(A + \epsilon_A)X = B + \epsilon_B$$

minimize $\sum \epsilon_A^2 + \epsilon_B^2$

$$\sum X_{i,} = 1 \forall i$$

$$X > 0$$

the elements of ϵ_A are NULL if the corresponding elements of A are NULL. A typically contains biomarker concentrations for several taxonomic groups, and B field measurements of the same biomarkers. X is then an estimate of the taxonomic composition of the field sample.

Usage

```
tlsce(A, B, Wa=NULL, Wb=NULL, minA=NULL, maxA=NULL,
       A_init=A, Xratios=TRUE, ...)
```

Arguments

`A`	a matrix or data frame. If A contains biomarker data for taxonomic groups, the biomarkers have to be organized per row, and the taxonomic groups per column.
`B`	a matrix or data frame. If B contains biomarker field data, the biomarkers have to be organized per row, and the samples per column.
`Wa`	weighting of A, a matrix with the same dimensions of A. If `Wa=NULL`, Wa defaults to 1. This parameter can be used to give more importance to elements of A or A in total compared to B. weights are implemented as proportional to $1/s$ (as opposed to $1/s^2$) with s the standard deviation of the error term.

Wb	weighting of B, a matrix with the same dimensions of B. If Wb=NULL, Wb defaults to 1. This parameter can be used to give more importance to elements of B or B in total compared to A. weights are implemented as proportional to $1/s$ (as opposed to $1/s^2$) with s the standard deviation of the error term.
minA	minimum values for A
maxA	maximum values for A
A_init	a matrix with the same structure as A. a general, non-linear optimization routine (default nlminb) is used to minimize the sum of squared residuals of A versus the fitted matrix A_fit (see falue). This optimization routine requires a set of starting values, by default the non-zero elements of A. This provides a good fit, but when in doubt about the convergence of the algorithm, one can provide different starting values for the optimization routine in A_init.
Xratios	TRUE or FALSE: are the colSums of the matrix X equal to 1? This is for example the case in a compositional matrix. (only if A and B are both expressed relative to the unit of biomass) if Xratios =TRUE, A has pigment concentrations per biomass unit, B has pigment concentrations per biomass unit per sample, and X contains ratios of biomass unit per sample. if Xratios =FALSE, A has pigment concentrations per biomass unit, B has pigment concentrations per sample, and X has biomass units per sample
...	Arguments to be passed to lsei() or to modFit()

Details

instead of a linear least squares regression, in which the elements of A would be fixed, the function tlsce includes the non-zero elements of A in the least squares regression. This is similar to other total least squares regression methods (also called orthogonal regression), with the main difference that only non-zero elements of A contain an error term.

Value

A list with the following elements:

X	Array with dimension c(ncol(A),ncol(B), iter) containing the species composition of each sample
A_fit	Array with same dimension as A, containing the best-fit values of the input biomarker data per taxonomic group
B_fit	Array with same dimension as B, containing the biomarker field data, corresponding to Afit
solutionNorms	a vector of 3 values: the value of the minimised quadratic function at the solution, in this case $\sum (Afit - A) * Wa)^2 + (Bfit$ and the shares of this value attributed to A and to B
convergence	An integer code. '0' indicates successful convergence.

Author(s)

Karel Van den Meersche <k.vdmeersche@nioo.knaw.nl>, Karline Soetaert <k.soetaert@nioo.knaw.nl>

References

Van den Meersche, K., K. Soetaert and J.J. Middelburg (2008) *A Bayesian compositional estimator for microbial taxonomy based on biomarkers*, Limnology and Oceanography Methods 6, 190-199

See Also

BCE

Examples

```
A <- t(bceInput$Rat)
B <- t(bceInput$Dat)
tlsce(A,B)
## weighting Wa inversely proportional to A
tlsce(A,B,Wa=1/A)
```

Index