Package ‘concordance’

November 14, 2024

Title Concordance Analysis and Active Subspaces for High-Dimensional Computer Models

Version 1.0.0

Description
Tools for estimating the Constantine Matrix for a computer model \`f\` (or alternatively, the co-Constantine Matrix for two functions \`f\` and \`g\`, as in a \`\`Concordance Analysis''). Works efficiently
in high-dimensions by leveraging analytic results based on the Bayesian MARS emulator (with the \`BASS\` package).

Imports lhs, BASS, zipfR

Depends R (>= 3.5.0)

Suggests knitr, rmarkdown, mvtnorm, testthat (>= 2.1.0)

License BSD_3_clause + file LICENSE

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Kellin Rumsey [aut, cre]

Maintainer Kellin Rumsey <`knrumsey@lanl.gov`>

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Contents

`act_dims` *Active Dimension (Not validated. Might be buggy)*

Description

This function estimates the dimensions of the active subspace using a sequential testing approach

Usage

```
act_dims(C, X, y, k = ncol(C), alpha = 0.05, all_sets = TRUE, verbose = TRUE)
```

Arguments

`C`	Constantines C matrix (e.g. from C_bass, C_mc, or C_gp)
`X`	the original input variables
`y`	the original response variable (mod$y when using C_bass(mod))
`k`	The maximum number of columns of W to consider
`alpha`	significance threshold for testing procedure
`all_sets`	should all dimension sets be returned? Or just the smallest set.
`verbose`	should progress be printed

Value

a list of active subspace dimensions

act_scores *Activity Scores*

Description

This function computes the activity scores for main effects of the variables

Usage

```
act_scores(C, k = 1, plt = FALSE, norm = FALSE)
```

Arguments

`C`	Constantines C matrix (e.g. from C_bass, C_mc, or C_gp)
`k`	The number of columns of W to consider
`plt`	Logical, should a plot be made?
`norm`	Logical, should activity scores be normalized to have length one?

Value

the activity scores

bassfunc2bass *Convert functional BASS model to BASS model*

Description

The argument to this function is the output of a `bass()` call when a single functional variable is specified using the `xx.func` argument. Note that the resulting model may not be a valid bass object for some applications, but the resulting model can be passed to `concordance::C_bass()` and related functions.

Usage

```
bassfunc2bass(bfm)
```

Arguments

`bfm`	an object of class bass, where a functional variable has been specified.

Examples

```
#The following are equivalent
n <- 100 #Number of observations
p <- 4   #Number of variables (beyond p = 2, variables are inert)
X <- matrix(runif(n*p), nrow=n)
y <- apply(X, 1, ff1)
gm <- gbass(X, Y, nmcmc=1000, nburn=901)
bm <- gm2bm(gm)
sob <- sobol(bm)
plot(sob)
```

`borehole_grad` *The Gradient of Borehole Function*

Description

This function returns the gradient of the borehole function.

Usage

```
borehole_grad(xx, design = 0.5, adjust = TRUE)
```

Arguments

`xx`	the 7 inputs, restricted to the unit interval
`design`	the radius of the borehole (typically rw)
`adjust`	logical. adjustment for scaling needed?

Details

PARAMETER RANGES rw in `[0.05, 0.15]` radius of borehole (m) r in `[100, 50000]` radius of influence (m) Tu in `[63070, 115600]` transmissivity of upper aquifer (m2/yr) Hu in `[990, 1110]` potentiometric head of upper aquifer (m) Tl in `[63.1, 116]` transmissivity of lower aquifer (m2/yr) Hl in `[700, 820]` potentiometric head of lower aquifer (m) L in `[1120, 1680]` length of borehole (m) Kw in `[9855, 12045]` hydraulic conductivity of borehole (m/yr)

Value

The output of the borehole function

`build_prior` *Build Prior Method for C_bass and Cfg_bass*

Description

A quick way to build priors for use in `C_bass` and `Cfg_bass`. For more complicated priors, such as mixture distributions, see details in `?C_bass`.

Usage

```
build_prior(
  dist,
  trunc = NULL,
  mean = NULL,
  sd = NULL,
  shape1 = NULL,
  shape2 = NULL,
  shape = NULL,
  scale = NULL
)
```

Arguments

`dist`	A vector of length p. Valid entries include "uniform", "normal", "beta", "gamma".
`trunc`	A matrix of dimension px2 (rows are recycled if nrow < p). `Inf` is a valid entry.
`mean`	A p-vector of means (used for normal only)
`sd`	A p-vector of sds (used for normal only)
`shape1`	A p-vector of shape1 parameters for beta prior
`shape2`	A p-vector of shape2 parameters for beta prior
`shape`	A p-vector of shape parameters for gamma prior
`scale`	A p-vector of scale parameters for gamma prior

Details

All vectors and matrix rows are recycled for parameters. The vector `dist` cannot be recyled as it defines p.

Value

a list which can be passed into C_bass or Cfg_bass as a prior.

`Cfg_bass` *Estimate Cfg with BASS*

Description

Closed form estimator of the Cfg matrix using a BASS model

Usage

```
Cfg_bass(mod1, mod2, prior = NULL, mcmc.use = NULL, scale01 = FALSE)
```

Arguments

`mod1`	a fitted BASS model for first function
`mod2`	a fitted BASS model for second function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a 2-column matrix with a pair of indices in each row.
`scale01`	logical (default FALSE). When TRUE, the the C matix corresponds to the (0, 1)-scaled inputs rather than the original inputs.

Details

prior should be a list of length p (one object for each variable). Each element of prior should be a named list with fields

dist - ("uniform", "normal").

trunc - truncation bounds (a, b)

mean - vector of means (mixture of normals only)

sigma - vector of sds (mixture of normals only)

weights - vector of mixture weights (mixture of normals only)

Value

A list representing the posterior distribution of the Co-Constantine matrix (Cfg).

`Cfg_bassPCA` *Estimate Cfg matrix with bassPCA as a function of t*

Description

Closed form estimator of the Cfg(t) matrix using a BASS model

Usage

```
Cfg_bassPCA(modPCA1, modPCA2, prior = NULL, mcmc.use = NULL, func.use = NULL)
```

Arguments

`modPCA1`	a fitted model of class bassBasis from bassPCA function
`modPCA2`	a fitted model of class bassBasis from bassPCA function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix
`func.use`	a vector of points of the functional variable to use

Details

This function works by converting the linear combination of bass models to a single bass model. See Cfg_bass for more details

Value

A list returning the (posterior samples?) of the Cfg matrix for each point specified in func.use

---

`Cfg_bassPCA_v2` *Estimate Cfg matrix with bassPCA as a function of t*

Description

Closed form estimator of the Cfg(t) matrix using a BASS model. An alternative approach, see details.

Usage

```
Cfg_bassPCA_v2(
  modPCA1,
  modPCA2,
  prior = NULL,
  mcmc.use = NULL,
  func.use = NULL
)
```

Arguments

`modPCA1`	a fitted model of class bassBasis from bassPCA function
`modPCA2`	a fitted model of class bassBasis from bassPCA function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix
`func.use`	a vector of points of the functional variable to use

Details

This function works by decomposing the Cfg of a linear combination into the pairwise Cfigj matrices of the components. See Cfg_bass for more details

Value

A list returning the (posterior samples?) of the Cfg matrix for each point specified in func.use

---

`Cfg_mc` *C_fg matrix with Monte Carlo*

Description

Approximates generalized C matrix with Monte Carlo for functions f and g

Usage

```
Cfg_mc(
  f,
  g,
  measure,
  grad = FALSE,
  nmc = 10000,
  names = NULL,
  seed = NULL,
  return_C = FALSE,
  ...
)
```

Arguments

`f`	the function f (or gradient of f, if grad=TRUE)
`g`	the function g (or gradient of f, if grad=TRUE)
`measure`	the number of inputs in f. See details for more sophisticated use (for non-uniform measure)
`grad`	if TRUE f is assumed to return the gradient of f. When FALSE, forward diff is used for gradient approximation.
`nmc`	the number of Monte Carlo replications
`names`	(optional) names for the functions f and g

`seed`	optional. seed for MC draws
`return_C`	(default FALSE). When TRUE, the object returned is a list with components Cf Cg Cfg
`...`	additional arguments passed to f()

Details

`measure` should be an argument-free function which simulates a draw x ~ p(x) where p is the prior measure. Alternatively, `measure` can be a numeric scalar, in which case the Monte Carlo draws are simulated from the standard uniform distribution as `runif(measure[1])`.

Value

the approximated C matrix

`coactive_bass` *Concordance Analysis (with BASS)*

Description

Computes the concordance between mod1 and mod2 (BASS models representing f1 and f2)

Usage

```
coactive_bass(mod1, mod2, prior = NULL, mcmc.use = NULL, q = 1, ...)
```

Arguments

`mod1`	BASS model representing function 1
`mod2`	BASS model representing function 2
`prior`	NULL (default) Uniform(0,1) prior for each variable. See details fr required prior structure.
`mcmc.use`	a vector of mcmc replications to use. Can also be a 2-column matrix with indices for f1 and f2.
`q`	order for the activity score measures
`...`	additional arguments passed fd_grad()

Details

`measure` should be an argument-free function which simulates a draw x ~ p(x) where p is the prior measure. If `measure` is numeric, then Monte Carlo draws are simulated from the standard uniform distribution as `runif(measure[1])`.

Value

Estimates of C1, C2, C12, V12, conc(f1, f2), contributions and coactivity scores

`coact_scores` *Co-Activity Scores*

Description

This function computes the activity scores for main effects of the variables

Usage

```
coact_scores(V, q = 1, signed = TRUE, plt = FALSE, norm = FALSE)
```

Arguments

`V`	The symmetrized co-Constantine matrix from Cfg_bass()
`q`	The number of columns of W to consider
`signed`	Use signed or unsigned version?
`plt`	Logical, should a plot be made?
`norm`	Logical, should activity scores be normalized to have a length of 1?

Value

the coactivityactivity scores

`conc_analysis_mc` *Concordance Analysis*

Description

Performs a full concordance analysis between f and g

Usage

```
conc_analysis_mc(
  f,
  g,
  measure,
  grad = FALSE,
  nmc = 10000,
  names = c("f", "g"),
  seed = NULL,
  ...
)
```

Arguments

`f`	the function f (or gradient of f, if grad=TRUE)
`g`	the function g (or gradient of g, if grad=TRUE)
`measure`	the number of inputs in f and g. See details for more sophisticated use (for non-uniform measure)
`grad`	if TRUE f and g are assumed to return gradients. When FALSE, forward diff is used for approximation.
`nmc`	the number of monte carlo replications
`names`	names of the functions
`seed`	optional. seed for monte carlo draws
`...`	additional arguments passed fd_grad()

Details

`measure` should be an argument-free function which simulates a draw x ~ p(x) where p is the prior measure. If `measure` is numeric, then Monte Carlo draws are simulated from the standard uniform distribution as `runif(measure[1])`.

Value

a list with components: C (constantine matrices), principle_grads, contributions, totals, conc, dist

---

`conc_bass` *Concordance analysis using bass models*

Description

Closed form estimator of the Cfg matrix using a BASS model

Usage

```
conc_bass(
  mod1,
  mod2,
  prior = NULL,
  mcmc.use = NULL,
  type = 1,
  prior_func = NULL,
  func.use = NULL
)
```

Arguments

`mod1`	a fitted BASS model for first function
`mod2`	a fitted BASS model for second function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix

`type`	Only used if class(mod1) == class(mod2) == "bassBasis". The default `type=1` calls `C_bassPCA_v2` and any other value calls `C_bassPCA`.
`prior_func`	Only used if class(mod1) == class(mod2) == "bassBasis". Optional weights for the prior on the functional variable.
`func.use`	Only used if class(mod1) == class(mod2) == "bassBasis".

Details

When models are class bass, each field of the returned object is a list for each mcmc iteration (or a vector for concordance). If mcmc.use = NULL or length(mcmc.use) = 1, then each field is just a matrix (or a scalar for concordance).

When models are class bassBasis (from bassPCA function), each field will be a list for each time point in func.use (func.use = NULL uses all time points in the training data by default). Each component of the list has the same structure as described above for the class == "bass" case.

Value

A list with matrices Cf, Cg, Cfg, Vfg and the concordance.

`conc_mc` *Concordance*

Description

Computes the concordance between f and g

Usage

```
conc_mc(f, g, measure, grad = FALSE, nmc = 10000, ...)
```

Arguments

`f`	the function f (or gradient of f, if grad=TRUE)
`g`	the function g (or gradient of g, if grad=TRUE)
`measure`	the number of inputs in f and g. See details for more sophisticated use (for non-uniform measure)
`grad`	if TRUE f and g are assumed to return gradients. When FALSE, forward diff is used for approximation.
`nmc`	the number of monte carlo replications
`...`	additional arguments passed fd_grad()

Details

`measure` should be an argument-free function which simulates a draw x ~ p(x) where p is the prior measure. If `measure` is numeric, then Monte Carlo draws are simulated from the standard uniform distribution as `runif(measure[1])`.

Value

the concordance between functions f and g

C_bass *Estimate the Constantine Matrix with BASS*

Description

Closed form estimator of the C matrix using a BASS model

Usage

```
C_bass(mod, prior = NULL, mcmc.use = NULL, scale01 = FALSE)
```

Arguments

`mod`	a fitted BASS model
`prior`	NULL (default) (0,1])prior for each variable. See details for required structure of prior
`mcmc.use`	set of indices telling which mcmc draws to use
`scale01`	logical (default FALSE). When TRUE, the the C matix corresponds to the (0, 1)-scaled inputs rather than the original inputs.

Details

prior should be a list of length p (one object for each variable). Each element of prior should be a named list with fields. See also the concordance::build_prior() function.

- dist - ("uniform", "normal", "beta", "gamma").
- trunc - truncation bounds (a, b). These should be c(0, 1) for "beta" and c(0, Inf) for "gamma".
- mean - vector of means (mixture of normals only)
- sd - vector of sds (mixture of normals only)
- shape1, shape2 - shape parameters for beta distribution
- shape, scale - parameters for gamma distribution
- weights - vector of mixture weights (currently only compatible with dist="normal")

Value

A list representing the posterior distribution of the Constantine matrix.

`C_bassPCA` *Estimate C matrix with bassPCA as a function of t*

Description

Closed form estimator of the C(t) matrix using a BASS model

Usage

```
C_bassPCA(modPCA, prior = NULL, mcmc.use = NULL, func.use = NULL)
```

Arguments

`modPCA`	a fitted model of class bassBasis from bassPCA function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix
`func.use`	a vector of points of the functional variable to use

Details

This function works by converting the linear combination of bass models to a single bass model. See C_bass for more details

Value

A list returning the (posterior samples?) of the C matrix for each point specified in func.use

`C_bassPCA_v2` *Estimate C matrix with bassPCA as a function of t*

Description

Closed form estimator of the C(t) matrix using a BASS model. An alternative approach see details. This approach is usually faster than the alternative.

Usage

```
C_bassPCA_v2(modPCA, prior = NULL, mcmc.use = NULL, func.use = NULL)
```

Arguments

`modPCA`	a fitted model of class bassBasis from bassPCA function
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix
`func.use`	a vector of points of the functional variable to use

Details

This function works by decomposing the C of a linear combination into the pairwise Cfg matrices of the components. See C_bass for more details

Value

A list returning the (posterior samples?) of the C matrix for each point specified in func.use

---

C_mc *C matrix with Monte Carlo*

Description

Approximates Constantine's C with Monte Carlo for a function f

Usage

```
C_mc(f, measure, grad = FALSE, nmc = 10000, seed = NULL, ...)
```

Arguments

`f`	the function f (or gradient of f, if grad=TRUE)
`measure`	the number of inputs in f. See details for more sophisticated use (for non-uniform measure)
`grad`	if TRUE f is assumed to return the gradient of f. When FALSE, forward diff is used for gradient approximation.
`nmc`	the number of Monte Carlo replications
`seed`	optional. seed for MC draws
`...`	additional arguments passed to f()

Details

`measure` should be an argument-free function which simulates a draw x ~ p(x) where p is the prior measure. If `measure` is numeric, then Monte Carlo draws are simulated from the standard uniform distribution as `runif(measure[1])`.

Value

the approximated C matrix

`fd_grad` *Forward diff function*

Description

Function for approximating the gradient of a function

Usage

```
fd_grad(f, x, h = 1e-12, ...)
```

Arguments

`f`	the function to find the gradient of
`x`	the input values
`h`	the tolerance
`...`	additional inputs to be passed to f

Value

The approximate gradient of f at x

`f_borehole` *The Borehole Function*

Description

This function models the flow of water through a borehole.

Usage

```
f_borehole(xx, design = 0.5)
```

Arguments

`xx`	the 7 inputs, restricted to the unit interval
`design`	the radius of the borehole (typically rw)

Details

PARAMETER RANGES rw in `[0.05, 0.15]` radius of borehole (m) r in `[100, 50000]` radius of influence (m) Tu in `[63070, 115600]` transmissivity of upper aquifer (m2/yr) Hu in `[990, 1110]` potentiometric head of upper aquifer (m) Tl in `[63.1, 116]` transmissivity of lower aquifer (m2/yr) Hl in `[700, 820]` potentiometric head of lower aquifer (m) L in `[1120, 1680]` length of borehole (m) Kw in `[9855, 12045]` hydraulic conductivity of borehole (m/yr)

Value

The output of the borehole function

`f_piston` *Piston Function*

Description

Piston function studied by Constantine in global sensitivity metrics paper

Usage

```
f_piston(x)
```

Arguments

`x`	7 inputs. See Constantine paper for details

Details

PARAMETER RANGES: measure <- function() res <- c(runif(1, 30, 60), runif(1, .005, .02), runif(1, .002, .01), runif(1, 1000, 5000), runif(1, 90000, 110000), runif(1, 290, 296), runif(1, 340, 360)) return(res)

Value

Time to fire for piston

`K_bassPCA` *Estimate K matrix with bassPCA*

Description

Closed form estimator of the K matrix using a BASS model

Usage

```
K_bassPCA(
  modPCA,
  type = 1,
  prior = NULL,
  prior_func = NULL,
  mcmc.use = NULL,
  func.use = NULL
)
```

Arguments

`modPCA`	a fitted model of class bassBasis from bassPCA function
`type`	1 or 2. Use C_bassPCA or C_bassPCA_v2?
`prior`	NULL (default) `[0, 1]` prior for each variable. See details for required structure of prior
`prior_func`	a vector of weights to use when summing over functional variable. Should be same length as func.use.
`mcmc.use`	vector of mcmc indices to be used for both models. Otherwise, a matrix
`func.use`	a vector of points of the functional variable to use

Details

This function works by converting the linear combination of bass models to a single bass model. See C_bass for more details

Value

A list returning the (posterior samples?) of the C matrix for each point specified in func.use

---

`lcbass2bass` *Convert a linear combination of BASS models to a single BASS model*

Description

A linear combination of BASS models is also a BASS model. This function takes a list of BASS models (all with the same data matrix xx.des) and returns the resulting linear combination as a new BASS model. One useful application of this function is to convert bassPCA to bass for a fixed time point. Does not currently work for bass models with functional or categorical inputs.

Usage

```
lcbass2bass(
  mod_list,
  weights = rep(1, length(mod_list)),
  yy = NULL,
  mcmc.use = NULL
)
```

Arguments

`mod_list`	A list of bass models.
`weights`	An optional vector of weights.
`yy`	The data vector. Optional, but useful for some bass object methods.
`mcmc.use`	set of indices telling which mcmc draws to use.

Examples

```
a <- 1
```

`modified_borehole` *A Modified Borehole Function*

Description

This function is for testing, it is a modified borehole function designed to have a more interesting active subspace

Usage

```
modified_borehole(xx, design = 0.5)
```

Arguments

`xx`	5 inputs, restricted to the unit interval. More inputs can be used but they are completely inert.
`design`	the radius of the borehole (typically rw)

Details

PARAMETER RANGES rw in `[0.05, 0.15]` radius of borehole (m) r in `[100, 50000]` radius of influence (m) Tu in `[63070, 115600]` transmissivity of upper aquifer (m2/yr) Hu in `[990, 1110]` potentiometric head of upper aquifer (m) Tl in `[63.1, 116]` transmissivity of lower aquifer (m2/yr) Hl in `[700, 820]` potentiometric head of lower aquifer (m) L in `[1120, 1680]` length of borehole (m) Kw in `[9855, 12045]` hydraulic conductivity of borehole (m/yr)

Value

The output of the borehole function

`plot.ConcordanceAnalysis` *Plotting Function for object of class ConcordanceAnalysis*

Description

Plotting Function for object of class ConcordanceAnalysis

Usage

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

Arguments

`x`	object of class "ConcordanceAnalysis"
`...`	arguments to be passed to individual plot functions

`plot_active_grad_k` *Plot components of the kth Principle Gradient*

Description

Plot components of the kth Principle Gradient

Usage

```
plot_active_grad_k(obj, k = 1, vnames = NULL, ...)
```

Arguments

`obj`	object of class "ConcordanceAnalysis"
`k`	which principle gradient is desired?
`vnames`	optional vector of variable names
`...`	additional arguments passed to barplot

`plot_contributions` *Plot contributions*

Description

Plots the contributions pi_f, pi_g, and pi_fg

Usage

```
plot_contributions(obj, ...)
```

Arguments

`obj`	object of class "ConcordanceAnalysis"
`...`	additional arguments passed to barplot

`plot_sensitivities` *Plot sensitivities*

Description

The sensitivity of variable j (wrt to f) is defined as sum_i=1^n pi_f(i) * delta_f(j,i) The definition is similar for g or for fg

Usage

```
plot_sensitivities(obj, vnames = NULL, ...)
```

Arguments

`obj`	object of class "ConcordanceAnalysis"
`vnames`	optional vector of variable names
`...`	additional arguments passed to barplot

`print.ConcordanceAnalysis` *Summary and Print functions*

Description

Prints a summary for an object of class "ConcordanceAnalysis"

Usage

```
## S3 method for class 'ConcordanceAnalysis'
print(x, ...)
```

Arguments

`x`	object of class "ConcordanceAnalysis"
`...`	Additional arguments (ignored)

`summary.ConcordanceAnalysis` *Summary and Print functions*

Description

Prints a summary for an object of class "ConcordanceAnalysis"

Usage

```
## S3 method for class 'ConcordanceAnalysis'
summary(object, ...)
```

Arguments

`object`	object of class "ConcordanceAnalysis"
`...`	Ignored

`tr` *Trace of a matrix*

Description

Shortcut for sum(diag(A))

Usage

```
tr(A)
```

Arguments

`A`	a matrix

Value

The trace of a matrix

`Z_bass` *Estimate the Expected Gradient with BASS*

Description

Closed form estimator of Z = E(gradient f)

Usage

```
Z_bass(mod, prior = NULL, mcmc.use = NULL, scale01 = FALSE)
```

Arguments

`mod`	a fitted BASS model
`prior`	NULL (default) (0,1])prior for each variable. See details for required structure of prior
`mcmc.use`	set of indices telling which mcmc draws to use
`scale01`	logical (ignored in current version)

Details

prior should be a list of length p (one object for each variable). Each element of prior should be a named list with fields. See also the concordance::build_prior() function.

- dist - ("uniform", "normal", "beta", "gamma").
- trunc - truncation bounds (a, b). These should be c(0, 1) for "beta" and c(0, Inf) for "gamma".
- mean - vector of means (mixture of normals only)
- sd - vector of sds (mixture of normals only)
- shape1, shape2 - shape parameters for beta distribution
- shape, scale - parameters for gamma distribution
- weights - vector of mixture weights (currently only compatible with dist="normal")

Value

A list representing the posterior distribution of the Constantine matrix.

Index