Package 'multivator'

May 29, 2017

Type Package

Title A Multivariate Emulator

Version 1.1-9

Depends R(>= 3.0.1)

Imports utils, emulator (>= 1.2-15), mvtnorm, methods

Suggests abind

Date 2017-05-29

Author Robin K. S. Hankin

Maintainer Robin K. S. Hankin <hankin.robin@gmail.com>

Description A multivariate generalization of the emulator package.

License GPL-2

LazyLoad yes

LazyData yes

NeedsCompilation no

Repository CRAN

Date/Publication 2017-05-29 03:39:52 UTC

R topics documented:

`multivator-package` *A multivariate emulator*

Description

A generalization of the emulator as discussed in Hankin 2005

Details

Package:	multivator
Type:	Package
Version:	1.0
Date:	2009-10-27
License:	GPL-2
LazyLoad:	yes

Author(s)

Robin K. S. Hankin

Maintainer: <hankin.robin@gmail.com>

References

R. K. S. Hankin 2005. "Introducing BACCO, an R bundle for Bayesian Analysis of Computer Code Output". *Journal of Statistical Software*, 14(16).

R. K. S. Hankin (2012). "Introducing multivator: A Multivariate Emulator" *Journal of Statistical Software*, 46(8), 1-20. http://www.jstatsoft.org/v46/i08/

See Also

`multem`

Examples

```
data(mtoys)
d <- obs_maker(toy_mm, toy_mhp, toy_LoF, toy_beta)

ex <- experiment(toy_mm,d)

multem(toy_mm2, ex, toy_mhp, toy_LoF,give=TRUE)
```

`apart` *Decompose a matrix with multiple columns of dependent variables*

Description

Decomposes a matrix with multiple columns of dependent variables into a `mdm` object

Usage

```
apart(X, dependent, use_rownames = TRUE)
```

Arguments

`X`	A matrix with columns corresponding to either independent variables or dependent variables. The names of the independent variables are taken from the column names of `X`
`dependent`	Vector of length `ncol(X)`. If numeric, interpret as the column numbers of the dependent variable. If logical, `TRUE` elements correspond to dependent variables
`use_rownames`	Boolean, with default `TRUE` meaning to use the rownames of `X` to create rownames in the returned value

Value

Returns an object of class `experiment`.

Author(s)

Robin K. S. Hankin

See Also

`as.list`

Examples

```
data(e3mg)
apart(e3mg , 6:7)

a <- round(emulator::latin.hypercube(6,5),2)
rownames(a) <- c("first","second","third","fourth","fifth","sixth")
colnames(a) <- c(letters[1:3],"length","depth")
jj_expt <- apart(a,4:5)    # use of apart()

x <- get_mdm(jj_expt[c(1,7)])
xold(x) <- 0.5

multem(x,jj_expt,hp=as.mhp(x),give=TRUE)
```

---

as.separate *Split an object of class* experiment *into a list of univariate datasets*

Description

Split an experiment object into univariate designs; return a list with elements suitable for univariate analysis with the emulator package.

Usage

```
as.separate(expt)
```

Arguments

expt	Object of class experiment

Author(s)

Robin K. S. Hankin

Examples

```
require(emulator)

data(mtoys)
d <- obs_maker(toy_mm, toy_mhp, toy_LoF, toy_beta)

ex <- experiment(toy_mm, d)
jj <- as.separate(ex)   #list of 3: temp,rain,humidity


# now use it in a univariate emulator:
kk <- jj$temp
```

```
interpolant.quick(x=latin.hypercube(3,4),d=kk$obs,xold=kk$val,scales=rep(1,4))
```

---

betahat *Various intermediate expressions needed by the multivariate emulator*

Description

Various intermediate expressions needed by the multivariate emulator

Usage

```
regressor(x,LoF)
beta_hat(expt,hp,LoF, ...)
betahat_mult(H, Sigmainv, d)
betahat_mult_Sigma(H, Sigma, d)
cstar(x1, x2=x1 , expt, hp,  LoF = NULL, Sigmainv=NULL, ...)
eq2.36(H, Sigmainv, d, log=TRUE)
eq2.36_Sigma(H, Sigma, d)
var.matrix(x1,x2=x1,hp, ...)
```

Arguments

x,x1,x2	Objects of class mdm: multivariate design matrix
H	Matrix of regressors (create this with regressor())
d	Vector of observations, possibly not all of the same dimensions (eg some elements might be Kelvin, others millimeters of rain per year)
expt	Object of class experiment
Sigma	The variance matrix of d
log	Boolean, with TRUE meaning to return the logarithm of the answer
Sigmainv	The inverse of the variance matrix of d, with default NULL meaning to calculate it directly using var.matrix()
LoF	A list of functions with default NULL meaning to use default_LoF()
hp	Object of class mhp: multivariate hyperparameters
...	Extra arguments which are passed (via var.matrix()) to corr.matrix() of the emulator package

Details

Function `regressor()` creates a (sort of) direct sum of regressor matrices for an overall regressor matrix. It returns a matrix whose rows are the regressor functions for each row in the `df` argument. Each type of observation has its own 'slot' of columns, the others being filled with zeros.

The emulator package *should* have used this method (rather than messing about with `regressor.basis()` and `regressor.multi()`).

To get the regression coefficients, the user should use function `beta_hat()`, which is the user-friendly version. It is a wrapper for function `betahat_mult_Sigma()`.

The equation for `var.matrix()` is

$$c^*\left(x,x'\right)=c\left(x,x'\right)-t(x)^TA^{-1}t(x')+\left\{h(x)^T-t(x)^TA^{-1}H\right\}\left(H^TA^{-1}H\right)^{-1}\left\{h(x')^T-t(x')^TA^{-1}H\right\}^T$$

Author(s)

Robin K. S. Hankin

See Also

multem

Examples

```
data(mtoys)

H <- regressor(toy_mm, toy_LoF)
Sigma <- var.matrix(toy_mm, hp=toy_mhp)
Sigmainv <- solve(Sigma)


jj <- toy_mm_maker(34,35,36)
expt  <- experiment(jj,obs_maker(jj,toy_mhp,toy_LoF,toy_beta))

x1 <- jj[c(20,40,100),]
xold(x1) <- 0.2

x2 <- jj[c(11,21:24,40:42),]
xold(x2) <- xold(x2)+0.1

#primary function of package:
multem(x=x1, expt, hp=toy_mhp, LoF=toy_LoF)

# conditional covariance matrix:
cstar(x1,x2, expt, hp=toy_mhp, LoF=toy_LoF)
```

`compatible` *Are two objects compatible?*

Description

Function to detect whether two objects are compatible

Usage

```
compatible(x1,x2)
```

Arguments

`x1,x2`	Two objects with `names` and `levels`. Typically either objects of class `mhp` or `mdm`.

Details

Here, "compatible" means have the same names and levels. If an `mdm` object and `mhp` object are compatible, then they may be supplied to (eg) `var.matrix()`.

The function uses `identical()` to compare the names and levels.

Value

Returns a Boolean.

Note

Cannot yet compare `LoF` objects.

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)
stopifnot(compatible(toy_mhp, toy_mm))
```

default_LoF *Default List of functions*

Description

Creates a default List of Functions for use with `regressor()`.

Usage

```
default_LoF(x)
```

Arguments

`x`	Object with `names` and `levels` methods; typically of class `mdm` or `mhp`.

Value

Returns a named list with each element giving the regressor functions for that level.

Author(s)

Robin K. S. Hankin

See Also

`regressor`

Examples

```
data(mtoys)

default_LoF(toy_mm)  # note list names == levels(toy_mm)

regressor(toy_mm)           # use default
regressor(toy_mm , toy_LoF) # use a bespoke set
```

e3mg *Output from computer model e3mg*

Description

Output from computer model e3mg detailing the depth of the recession and its length as a function of four exogenous parameters

Usage

```
data(e3mg)
```

Format

- e3mg is a matrix with 843 rows and 6 columns. Four of the columns are exogenous variables (oil.price, direct.tax, interest.rate, and saving.ratio) and two are model outputs: rec_len, the length (in years) of the recession, and dep_rec, the depth of the recession.
- e3mg_LoF is a list of functions suitable for use with the e3mg dataset

Details

The data comprises 843 runs of the e3mg econometric model, used to predict the recession precipitated by the banking crisis.

The depth of the recession is defined as the maximum difference between predicted post-crash GDP and GDP immediately pre-crash.

The length of the recession is defined as the time in years required for GDP to return to pre-crash levels.

Source

Data kindly provided by Cambridge Econometrics

See Also

apart

Examples

```
data(e3mg)
a <- lm(rec_len~oil.price*direct.tax + direct.tax*saving.ratio + investment,data=data.frame(e3mg))
b <- lm(rec_dep~oil.price*direct.tax + direct.tax*saving.ratio + investment,data=data.frame(e3mg))
plot(residuals(a),residuals(b))  # correlated!



# define an experiment object and find optimal prarams
e3mg_expt <- apart(e3mg[1:20,],6:7)
opt <- optimal_params(e3mg_expt, e3mg_LoF, option='c')


# now a point in parameter space:
center <- get_mdm(e3mg_expt)[c(1,40),]
rownames(center) <- c('center_dep','center_len')
xold(center) <- 0


#now predict the behaviour at the center:
multem(center, e3mg_expt, hp=opt, e3mg_LoF, give = TRUE)
```

`experiment` *Multivatriate hyperparameter (mhp) objects*

Description

Create and manipulate multivatriate hyperparameter (mhp) objects

Usage

```
experiment(mm,obs)
```

Arguments

`mm`	Object of class `mdm`
`obs`	Vector of observations, with elements corresponding to the rows of `mm`

Details

An "experiment" is an ordered pair of a multivariate design matrix and a vector of observations with entries corresponding to the rows of the design matrix.

It functions as a container for the design matrix and observations. It is intended to simplify the calls to many functions in the package which require a design matrix and vector of observations.

There are two get methods, `get_mdm()` and `get_obs()`, for the design matrix and observations respectively. Note the deliberate absence of set methods.

Value

Returns an object of class `experiment`, which is used as input to many of the functions in the package.

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)
jj_expt <- experiment(toy_mm,toy_d)


# accessor methods:
get_obs(jj_expt)
get_mdm(jj_expt)


# estimation of coefficients:
beta_hat(jj_expt, toy_mhp, toy_LoF)
```

```
# use multem():
multem(toy_mm3, jj_expt, toy_mhp, toy_LoF,give=TRUE)
```

---

head *Head and tail*

Description

Print the first few, or last few, lines of a mdm object

Usage

```
## S4 method for signature 'mdm'
head(x, n = 6, ...)
## S4 method for signature 'mdm'
tail(x, n = 6, ...)
```

Arguments

x	object of class mdm
n	number of lines to print as per same argument in head() and tail()
...	Further arguments passed to head() or tail()

Value

Returns a truncated mdm object. The levels of the types are unchanged.

Author(s)

Robin K. S. Hankin

Examples

```
data("mtoys")

head(toy_mm)
tail(toy_mm,3)
```

ipd *Positive definite matrices*

Description

Is a matrix symmetric positive-definite?

Usage

```
ipd(mat)
```

Arguments

mat	A matrix

Value

Returns either TRUE if symmetric positive-definite; or FALSE, printing a diagnostic message.

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)
stopifnot(ipd(crossprod(matrix(rnorm(30),10))))
stopifnot(ipd(M(toy_mhp)))
```

mcneall *Dataset due to McNeall*

Description

Data, due to McNeall, from 92 runs of a climate model

Usage

```
data(mcneall)
```

Details

McNeall used a numerical climate model and ran it 92 times, on a design matrix specified on 16 independent variables as detailed in McNeall 2008.

The model output is a temperature distribution over the surface of the Earth. The model gives 2048 temperatures, corresponding to 2048 grid squares distributed over the Earth. A vector of 2048 temperatures may be displayed on a global map using the `showmap()` function.

The 92 model runs are presented in the form of a 2048 by 92 matrix `mcneall_temps`, each column of which corresponds to a run. A row of 92 temperatures corresponds to the temperature at a particular place on the earth as predicted by each of the 92 model runs.

Following McNeall, a principal component analysis on the maps was performed. The first four were used. Matrix `eigenmaps` is a 2048 by 4 matrix, with columns corresponding to the four principal components.

Matrix `mcneall_pc` is a 92-by-20 matrix. The first 16 columns correspond to the independent variables (ie the design matrix); columns 17-20 correspond to the first four principal components of the model output. The 92 rows correspond to the 92 model runs.

The package can be used on the `mcneall_temps` matrix; use `apart()` to generate a `mdm` object. A reasonably optimized hyperparameters object of class `mhp` is given as `opt_mcneall`.

References

D. McNeall 2008. "Dimension Reduction in the Bayesian analysis of a numerical climate model". PhD thesis, University of Southampton.

See Also

showmap

Examples

```
data(mcneall)

showmap(mcneall_temps[,1], pc=FALSE,landmask=landmask)
```

`mdm` *Multivariate design matrices*

Description

Multivariate design matrices are represented using objects of class `mdm`.

Usage

```
mdm(xold, types)
as.mdm(x, ...)
is.mdm(x)
as.list(x, ...)
as.matrix(x, ...)
## S4 method for signature 'mdm,missing,missing'
as.data.frame(x, row.names=NULL,optional=TRUE, ...)
## S4 method for signature 'mdm'
rbind(x, ..., deparse.level=1)
types(x)
xold(x)
```

Arguments

xold	Matrix of design points, each row being a point in parameter space
types	A factor holding the types of each observation
x	An object of class mdm
row.names,optional	Currently ignored
...	Further arguments passed to NextMethod()
deparse.level	As for rbind()

Details

Various functionality for creating and manipulating objects of class mdm (Multivariate Design Matrix).

Note

The internal representation has two slots, one for the design matrix proper (a matrix), and one for the types of observation (a factor).

Author(s)

Robin K. S. Hankin

See Also

mhp,apart

Examples

```
mm <- toy_mm_maker(7,8,9)
is.mdm(mm)
xold(mm) <- matrix(rnorm(108),27,4)
mm[1,1] <- 0.3
```

```
data(mtoys)
obs_maker(mm,toy_mhp,toy_LoF,toy_beta)
```

---

mhp *Multivatriate hyperparameter (mhp) objects*

Description

Create and manipulate multivatriate hyperparameter (mhp) objects

Usage

```
mhp(M, B, levels = NULL, names = NULL)
is.mhp(x)
M(x)
M(x) <- value
B(x)
B(x) <- value
levels(x)
summary(object,...)
```

Arguments

M	Variance matrix (must be positive definite)
B	Array of roughness parameters. Each slice (ie B[,,i]) must be positive-definite
levels	Character vector holding the levels. Default NULL means to use rownames(M) or dimnames(B[[3]])
names	Character vector holding the names of the dimensions. Default of NULL means to use dimnames(B[[1]])
x,object	Object of class mhp
value	Replacement object
...	Further arguments passed to the summary method

Details

An mhp object *must* have names and levels, so either provide them explicitly with the eponymous arguments, or give named arrays to M and B.

Value

Returns an object of class mhp

Author(s)

Robin K. S. Hankin

See Also

mdm

Examples

```
hp <- mhp(M=diag(2),B=array(c(diag(3),diag(3)),c(3,3,2)),
names=letters[1:3],levels=c("oak","ash"))
M(hp)
B(hp)[1,1,1] <- 30  # try a negative value and see what happens
names(hp)
names(hp) <-  c("Alice","Zachy","Annabel")
levels(hp) <- c("squid","snail")
summary(hp)
```

---

mtoys *Toy datasets*

Description

Toy datasets that illustrate the package

Usage

```
toy_LoF
toy_mm
toy_mm2
toy_mm3
toy_mhp
```

Format

- toy_LoF is a list of three functions that work with regressor() and toy_df
- toy_M is an example M matrix for use with mhp()
- toy_B is an example of a B array of roughness coefficients for use with mhp()
- toy_mm and toy_mm2 are examples of a mdm object, generated with function toy_mm_maker(). These objects are marginals from the *same* multivariate observation.
- toy_mm3 and toy_mm4 are small examples of mdm objects
- toy_mhp is an example of a mhp object
- toy_beta is a numeric vector that works with the above objects

Details

These objects are intended as simple working 'toy' examples of the various things needed to use the emulator.

Note that toy_d and toy_d2 are the marginals of the *same* observation (see the vignette).

Author(s)

Robin K. S. Hankin

References

- R. K. S. Hankin 2005. *Introducing BACCO, an R bundle for Bayesian analysis of computer code output*, Journal of Statistical Software, 14(16)

See Also

toy_mm_maker

Examples

```
data(mtoys)
obs_maker(toy_mm, toy_mhp, toy_LoF, toy_beta)

multem(toy_mm2,toy_expt,toy_mhp,toy_LoF,give=TRUE)
```

---

multem *The multivariate emulator*

Description

A multivariate generalization of the `interpolant()` function of the `emulator` package

Usage

```
multem(x, expt, hp,  LoF = NULL, give=FALSE, Sigmainv=NULL, ...)
```

Arguments

`x`	Points at which the function is to be estimated in the form of an object of class `mdm`
`expt`	Points at which the code has been evaluated (`x_known`), in the form of an object of class `experiment`
`hp`	hyperparameter object, of class `mhp`
`give`	Boolean, with `TRUE` meaning to return extra information and default `FALSE` meaning to return just the mean
`Sigmainv`	The inverse of the variance matrix of the observations with default `NULL` meaning to calculate using `var.matrix()`
`LoF`	List of regressor functions
`...`	Further arguments passed to `var.matrix()`

Details

This is the central function of the package. It is the analogue of `interpolant()` of the emulator package.

Author(s)

Robin K. S. Hankin

See Also

`betahat_mult`

Examples

```
data(mtoys)
d <- obs_maker(toy_mm, toy_mhp, toy_LoF, toy_beta)
ex <- experiment(toy_mm , d)

Sigmainv <- solve(var.matrix(toy_mm,hp=toy_mhp))
multem(x=toy_mm2, expt=ex, hp=toy_mhp,LoF=toy_LoF, give=TRUE)
```

---

obs_maker *Create observations*

Description

A function to create observations using known parameters and hyperparameters

Usage

```
obs_maker(x, hp, LoF, beta, Sigma=NULL, ...)
```

Arguments

`x`	Object of class `mdm`: each row is a point in parameter space
`hp`	Object of class `mhp`
`LoF`	List of functions
`beta`	Vector of regression coefficients
`Sigma`	Variance matrix, with default `NULL` meaning to use `var.matrix(x,hp)`
`...`	Further arguments passed to `var.matrix()`

Details

Uses the `mvtnorm` package to generate observations directly from the parameters and hyperparameters as a Gaussian process.

Value

Returns a (named) vector of observations. Note that the observations may have different units (eg temperature in Kelvin, rainfall in millimeters per year).

Author(s)

Robin K. S. Hankin

See Also

toy_mm_maker

Examples

```
data(mtoys)
d <- obs_maker(toy_mm , toy_mhp, toy_LoF, toy_beta)
d <- obs_maker(toy_mm_maker(6,7,8) , toy_mhp, toy_LoF, toy_beta)
```

---

optimal_params *Optimization of the hyperparameters*

Description

Optimization of the hyperparameters using a sequence of subfunctions.

Usage

```
optimal_params     (expt, LoF, start_hp, option = "a", ...)
optimal_B          (expt, LoF, start_hp, option = "a", verbose=FALSE, ...)
optimal_identical_B(expt, LoF, start_hp, verbose=FALSE, ...)
optimal_diag_M     (expt, LoF, start_hp)
optimal_M          (expt, LoF, start_hp, ...)
```

Arguments

expt	Object of class experiment
LoF	List of functions
start_hp	Start value for the hyperparameters, an object of class mhp. The various optimization routines use the different parts of start_hp as start points, and incrementally update it
option	In function optimal_B() and consequently optimal_params(), a character indicating whether to allow the scales to differ or not. • Default option "a" is the simplest: each univariate B matrix is a multiple of the identity matrix.

- Option "b" allows the B matrices to be any (positive definite) diagonal matrix.
- Option "c" specifies that B[,,j] is diagonal for each j and furthermore that B[i,i,1]=B[i,i,2]=...=B[i,i,r]. This option calls optimal_identical_B().

verbose — In function optimal_B(), Boolean with TRUE meaning to print debugging information and default FALSE meaning not to print anything

... — Further arguments passed to the optimization routine

Details

The user-friendly wrapper function is optimal_params(). This calls function optimal_B() first, as most of the analysis is conditional on B. Then optimal_diag_M() is called; this places the maximum likelihood estimate for σ^2 on the diagonal of M. Finally, optimal_M() is called, which assigns the off-diagonal elements of M.

Each of the subfunctions returns an object appropriate for insertion into a mhp object.

The "meat" of optimal_params() is

```
     B(out)  <- optimal_B     (mm, d, LoF, start_hp=out, option=option, ...)
diag(M(out)) <- optimal_diag_M(mm, d, LoF, start_hp=out, ...)
     M(out)  <- optimal_M     (mm, d, LoF, start_hp=out, ...)
return(out)
```

See how object out is modified sequentially, it being used as a start point for the next function.

Value

Returns a mhp object.

Note

Function optimal_diag_M() uses MLEs for the diagonals, but using each type of observation separately. It is conceivable that there is information that is not being used here.

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)

optimal_params(toy_expt,toy_LoF,toy_mhp,option='c',control=list(maxit=1))
```

Print *Methods for printing mhp and mdm objects*

Description

Methods for printing nicely

Usage

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

Arguments

x	An object of class mdm or mhp
...	Further arguments (currently ignored)

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)
a <- as.mhp(toy_mm)
a
```

showmap *Function to plot the McNeall dataset*

Description

A small wrapper function to plot a global map of temperature, which is useful when analyzing the McNeall dataset

Usage

```
showmap(z, pc, landmask, ...)
```

Arguments

z	A vector of length 2048 corresponding to temperatures on the Earth's surface
pc	Boolean, with TRUE meaning to interpret z as a principal component and FALSE meaning to interpret z as a temperature map
landmask	A matrix of zeros and ones corresponding to the Earth's surface with zero indicating sea and one indicating land; use data(mcneall)
...	Further arguments passed to filled.contour()

Author(s)

Robin K. S. Hankin

See Also

mcneall

Examples

```
data(mcneall)
showmap(mcneall_temps[,1],pc=FALSE,landmask=landmask)
```

---

ss *Overall variance matrix*

Description

Calculates the maximum correlations possible consistent with the roughness parameters

Usage

```
ss(A, B, Ainv, Binv)
ss_matrix(hp,useM=TRUE)
ss_matrix_simple(hp,useM=TRUE)
```

Arguments

A,B	Positive-definite matrices (roughness parameters)
Ainv,Binv	The inverses of A and B; if missing, compute explicitly
hp	An object of class mhp
useM	Boolean, with default TRUE meaning to multiply (pointwise) by M and FALSE meaning not to (so giving the maximum correlation consistent with the roughness matrices B)

Details

Function ss() calculates the maximum possible correlation between observations of two Gaussian processes at the same point (equation 24 of the vignette):

$$\left|\left(\frac{1}{2}B_r+\frac{1}{2}B_s\right)\left(\frac{1}{2}B_r^{-1}+\frac{1}{2}B_s^{-1}\right)\right|^{-1/4}$$

Functions ss_matrix() and ss_matrix_simple() calculate the maximum covariances among the types of object specified in the hp argument, an object of class mhp. Function ss_matrix() is the preferred form; function ss_matrix_simple() is a less efficient, but more transparent, version. The two functions should return identical output.

Value

Function ss() returns a scalar, ss_matrix() a matrix of covariances.

Note

Thanks to Stephen Stretton for a crucial insight here

Author(s)

Robin K. S. Hankin

Examples

```
data(mtoys)
ss_matrix(toy_mhp)
```

---

toy_mm_maker *Make a toy mm object*

Description

Create a toy mhp object with three levels: temperature, rainfall, and humidity.

Usage

```
toy_mm_maker(na, nb, nc, include_first = TRUE)
```

Arguments

na,nb,nc	Numbers of observations for each level
include_first	Boolean, with default TRUE meaning to include an extra observation of each level at the midpoint of the domain

Value

Returns an object of class `mhp`.

Author(s)

Robin K. S. Hankin

Examples

```
toy_mm_maker(4,5,6,FALSE)
toy_mm_maker(1,1,2,TRUE)
```

Index