

# Package ‘multivator’

February 20, 2015

**Type** Package

**Title** A multivariate emulator

**Version** 1.1-4

**Depends** R(>= 2.10.0)

**Imports** utils, emulator (>= 1.2-13), mvtnorm, methods

**Suggests** abind

**Date** 2011-07-28

**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** 2013-12-05 20:13:57

## R topics documented:

multivator-package . . . . .	2
apart . . . . .	3
as.separate . . . . .	4
betahat . . . . .	5
compatible . . . . .	7
default_LoF . . . . .	8
e3mg . . . . .	8
experiment . . . . .	10
head . . . . .	11
ipd . . . . .	12
mcneall . . . . .	12
mdm . . . . .	13

mhp . . . . .	15
mtoys . . . . .	16
multem . . . . .	17
obs_maker . . . . .	18
optimal_params . . . . .	19
Print . . . . .	21
showmap . . . . .	22
ss . . . . .	22
toy_mm_maker . . . . .	24
<b>Index</b>	<b>25</b>

---

multivator-package	<i>A multivariate emulator</i>
--------------------	--------------------------------

---

**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**

<code>X</code>	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 <code>X</code>
<code>dependent</code>	Vector of length <code>ncol(X)</code> . If numeric, interpret as the column numbers of the dependent variable. If logical, <code>TRUE</code> elements correspond to dependent variables
<code>use_rownames</code>	Boolean, with default <code>TRUE</code> meaning to use the rownames of <code>X</code> 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

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

## Author(s)

Robin K. S. Hankin

## See Also

[mulitem](#)

## 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:
mulitem(x=x1, expt, hp=toy_mhp, LoF=toy_LoF)

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

---

compatible	<i>Are two objects compatible?</i>
------------	------------------------------------

---

**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	<i>Default List of functions</i>
-------------	----------------------------------

---

**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	<i>Output from computer model e3mg</i>
------	--

---

**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 prams
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***Multivariate hyperparameter (mhp) objects*

---

**Description**

Create and manipulate multivariate hyperparameter (mhp) objects

**Usage**

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

**Arguments**

mm	Object of class <code>mdm</code>
obs	Vector of observations, with elements corresponding to the rows of <code>mm</code>

**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

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

### 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 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)
```

---

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, ...)
names(x) <- value
## 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 <code>mdm</code>
value	Replacement value
row.names, optional	Currently ignored
...	Further arguments passed to <code>NextMethod()</code>
deparse.level	As for <code>rbind()</code>

**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	<i>Multivariate hyperparameter (mhp) objects</i>
-----	--

---

## Description

Create and manipulate multivariate 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 <code>rownames(M)</code> or <code>dimnames(B[[3]])</code>
names	Character vector holding the names of the dimensions. Default of NULL means to use <code>dimnames(B[[1]])</code>
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 <code>mdm</code>
expt	Points at which the code has been evaluated ( <code>x_known</code> ), in the form of an object of class <code>experiment</code>
hp	hyperparameter object, of class <code>mhp</code>
give	Boolean, with <code>TRUE</code> meaning to return extra information and default <code>FALSE</code> meaning to return just the mean
Sigmainv	The inverse of the variance matrix of the observations with default <code>NULL</code> meaning to calculate using <code>var.matrix()</code>
LoF	List of regressor functions
...	Further arguments passed to <code>var.matrix()</code>

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

**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)
```

**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. <ul style="list-style-type: none"> <li>• Default option “a” is the simplest: each univariate B matrix is a multiple of the identity matrix.</li> <li>• Option “b” allows the B matrices to be any (positive definite) diagonal matrix.</li> <li>• Option “c” specifies that <math>B[, , j]</math> is diagonal for each <math>j</math> and furthermore that <math>B[i, i, 1]=B[i, i, 2]=\dots=B[i, i, r]</math>. This option calls optimal_identical_B().</li> </ul>
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  $\sigma^2$  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**

<code>x</code>	An object of class <code>mdm</code> or <code>mhp</code>
<code>...</code>	Further arguments (currently ignored)

**Author(s)**

Robin K. S. Hankin

**Examples**

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

---

showmap	<i>Function to plot the McNeall dataset</i>
---------	---

---

**Description**

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

**Usage**

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

**Arguments**

<code>z</code>	A vector of length 2048 corresponding to temperatures on the Earth's surface
<code>pc</code>	Boolean, with TRUE meaning to interpret <code>z</code> as a principal component and FALSE meaning to interpret <code>z</code> as a temperature map
<code>...</code>	Further arguments passed to <code>filled.contour()</code>

**Author(s)**

Robin K. S. Hankin

**See Also**

[mcneall](#)

**Examples**

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

---

ss	<i>Overall variance matrix</i>
----	--------------------------------

---

**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

<code>A,B</code>	Positive-definite matrices (roughness parameters)
<code>Ainv,Binv</code>	The inverses of <code>A</code> and <code>B</code> ; if missing, compute explicitly
<code>hp</code>	An object of class <code>mhp</code>
<code>useM</code>	Boolean, with default <code>TRUE</code> meaning to multiply (pointwise) by $M$ and <code>FALSE</code> 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	<i>Make a toy mm object</i>
--------------	-----------------------------

---

**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

## \*Topic **datasets**

e3mg, 8  
mcneall, 12  
mtoys, 16

## \*Topic **package**

multivator-package, 2  
[,experiment-method (experiment), 10  
[,mdm-method (mdm), 13  
[<-,mdm-method (mdm), 13

apart, 3, 9, 14

as.data.frame (mdm), 13

as.data.frame,experiment,ANY,ANY-method  
(experiment), 10

as.data.frame,experiment-method  
(experiment), 10

as.data.frame,mdm,missing,missing-method  
(mdm), 13

as.data.frame,mhp-method (mdm), 13

as.list, 3

as.list (mdm), 13

as.list,mdm-method (mdm), 13

as.matrix (mdm), 13

as.matrix,mdm-method (mdm), 13

as.mdm (mdm), 13

as.mdm,mhp-method (mdm), 13

as.mhp (mhp), 15

as.mhp,experiment-method (mhp), 15

as.mhp,mdm-method (mhp), 15

as.separate, 4

B (mhp), 15

B<- (mhp), 15

beta\_hat (betahat), 5

betahat, 5

betahat\_mult, 18

betahat\_mult (betahat), 5

betahat\_mult\_Sigma (betahat), 5

compatible, 7

cstar (betahat), 5

default\_LoF, 8

dim,mdm-method (mdm), 13

e3mg, 8

e3mg\_LoF (e3mg), 8

eigenmaps (mcneall), 12

eq2.36 (betahat), 5

eq2.36\_Sigma (betahat), 5

experiment, 10

experiment-class (experiment), 10

get\_mdm (experiment), 10

get\_obs (experiment), 10

head, 11

head,experiment-method (experiment), 10

head,mdm-method (head), 11

ipd, 12

is.mdm (mdm), 13

is.mhp (mhp), 15

landmask (mcneall), 12

levels (mhp), 15

levels,experiment-method (mdm), 13

levels,mdm-method (mdm), 13

levels,mhp-method (mhp), 15

levels<-,mdm-method (mdm), 13

levels<-,mhp-method (mhp), 15

M (mhp), 15

M<- (mhp), 15

mcneall, 12, 22

mcneall\_pc (mcneall), 12

mcneall\_temps (mcneall), 12

mdm, 13, 16

mdm-class (mdm), 13

mean\_temp (mcneall), 12

mhp, 14, 15

mhp-class (mhp), 15  
 mtoys, 16  
 multem, 2, 6, 17  
 multivator (multivator-package), 2  
 multivator-package, 2  
  
 names (mhp), 15  
 names,mdm-method (mdm), 13  
 names,mhp-method (mhp), 15  
 names<- (mdm), 13  
 names<- ,mdm-method (mdm), 13  
 names<- ,mhp-method (mhp), 15  
 ncol,mdm-method (mdm), 13  
 nrow,mdm-method (mdm), 13  
  
 obs\_maker, 18  
 opt\_mcneall (mcneall), 12  
 optimal\_B (optimal\_params), 19  
 optimal\_diag\_M (optimal\_params), 19  
 optimal\_identical\_B (optimal\_params), 19  
 optimal\_M (optimal\_params), 19  
 optimal\_params, 19  
  
 Print, 21  
 print.experiment (experiment), 10  
 print.mdm (Print), 21  
 print.mhp (Print), 21  
 print.mhpSummary (mhp), 15  
  
 rbind (mdm), 13  
 rbind,mdm-method (mdm), 13  
 regressor, 8  
 regressor (betahat), 5  
 rownames,mdm-method (mdm), 13  
 rownames<- ,mdm-method (mdm), 13  
  
 show,mdm-method (Print), 21  
 show,mhp-method (Print), 21  
 showmap, 13, 22  
 ss, 22  
 ss\_matrix (ss), 22  
 ss\_matrix\_simple (ss), 22  
 summary (mhp), 15  
 summary,mhp-method (mhp), 15  
  
 tail (head), 11  
 tail,experiment-method (experiment), 10  
 tail,mdm-method (head), 11  
 toy\_B (mtoys), 16  
 toy\_beta (mtoys), 16  
 toy\_d (mtoys), 16  
 toy\_d2 (mtoys), 16  
 toy\_expt (mtoys), 16  
 toy\_LoF (mtoys), 16  
 toy\_M (mtoys), 16  
 toy\_mhp (mtoys), 16  
 toy\_mm (mtoys), 16  
 toy\_mm2 (mtoys), 16  
 toy\_mm3 (mtoys), 16  
 toy\_mm4 (mtoys), 16  
 toy\_mm\_maker, 17, 19, 24  
 toy\_point (mtoys), 16  
 types (mdm), 13  
 types,mdm-method (mdm), 13  
 types,mhp-method (mhp), 15  
 types<- (mdm), 13  
 types<- ,mdm-method (mdm), 13  
 types<- ,mhp-method (mhp), 15  
  
 var.matrix (betahat), 5  
  
 xold (mdm), 13  
 xold,mdm-method (mdm), 13  
 xold<- (mdm), 13  
 xold<- ,mdm-method (mdm), 13