Package ‘PerMallows’

April 29, 2017

Type Package
Title Permutations and Mallows Distributions
Version 1.13
Date 2017-04-28
Author Ekhine Irurozki <ekhine.irurozqui@ehu.eus>, Borja Calvo <borja.calvo@ehu.eus>, Jose A. Lozano <ja.lozano@ehu.eus>
Maintainer Ekhine Irurozki <ekhine.irurozqui@ehu.eus>
Description Includes functions to work with the Mallows and Generalized Mallows Models. The considered distances are Kendall's-tau, Cayley, Hamming and Ulam and it includes functions for making inference, sampling and learning such distributions, some of which are novel in the literature. As a by-product, PerMallows also includes operations for permutations, paying special attention to those related with the Kendall's-tau, Cayley, Ulam and Hamming distances. It is also possible to generate random permutations at a given distance, or with a given number of inversions, or cycles, or fixed points or even with a given length on LIS (longest increasing subsequence).
License GPL (>= 2)
Depends Rcpp (>= 0.10.3)
Imports utils
LinkingTo Rcpp
RoxygenNote 6.0.1
NeedsCompilation yes
Repository CRAN
Date/Publication 2017-04-28 23:19:26 UTC

R topics documented:

compose .................................................. 2
count.perms ................................................. 3
cycle2str .................................................. 4
cycles2perm ............................................... 4
**Compose permutations**

**Description**

This function composes two given permutations. One of the arguments can be a collection of permutations, but not both at the same time. In this case, every permutation in the collection is composed with the other argument.

**Usage**

```r
compose(perm1, perm2)
```
count.perms

Arguments

perm1  a permutation or a collection of permutations
perm2  a permutation or a collection of permutations

Value

The composition of the permutations

Examples

compose(c(3,1,2,4), c(4,1,3,2))

count.perms

Count permutations at a distance

Description

Given a distance (kendall, cayley, hamming or ulam), the number of items in the permutations perm.length and distance value d, how many permutations are there at distance d from any permutation? It can be used to count the number of derangements and the permutations with k cycles (Stirling numbers of the first kind)

Usage

count.perms(perm.length, dist.value, dist.name = "kendall", disk = FALSE)

Arguments

perm.length  number of items in the permutations
dist.value    the distance
dist.name     optional. One of: kendall (default), cayley, hamming, ulam
disk          optional can only be true if counting the permutations at each Ulam distance. Instead of generating the whole set of SYT and count of permutations per distance, it loads the info from a file in the disk

Value

The number of permutations at the given distance
Examples

   count.perms(4,2,"kendall")
   count.perms(4,2,"ulam")
   count.perms(4,2,"hamming")
   count.perms(4,2,"cayley")
   # The number of derangements of length 6 is computed as follows
   len <- 6
   count.perms(perm.length = len, dist.value = len, dist.name = "h")
   # The number of permutations with one cycle is computed as follows
   num.cycles <- 1
   count.perms(perm.length = len, dist.value = len - num.cycles, dist.name = "c")

---

cycle2str

**Friendly display the cycles**

Description

Given a list with the cycles of a permutation, displays them in the standard cycle notation

Usage

cycle2str(cy)

Arguments

cy a list with the set of cycles

Examples

cycle2str(perm2cycles(c(1,2,3,4)))

---

cycles2perm

**Get the permutation given the cycles**

Description

Get the permutation as a vector given the set of cycles in which it factorizes

Usage

cycles2perm(cycles)

Arguments

cycles a list with the set of disjoint cycles
Value

The permutation in vector notation

Examples

cycles2perm(perm2cycles(c(1,5,2,3,4)))

---

Sample of permutations APA

Description

A rda file containing a sample of permutations of the American Psychology Association

Format

Each row is a permutation

---

Sample of permutations

Description

A rda file containing a sample of permutations

Format

Each row is a permutation

---

Get a permutation consistent with a decomposition vector

Description

Given a distance decomposition vector and a distance name, generate uniformly at random a permutation consistent with the decomposition vector.

Usage

decomp2perm(vec, dist.name = "kendall")
Arguments

vec the permutation
dist.name optional the name of the distance. One of: kendall (default), cayley, hamming

Value

The distance decomposition vector of the given permutation and distance

Examples

decomp2perm(c(1,0,1,0,0), "kendall")
decomp2perm(c(1,0,1,0,0), "cayley")
decomp2perm(c(1,0,1,0,0), "hamming")

dgmm Calculate the probability of a permutation in a GMM

Description

Calculate the probability of a permutation sigma in a GMM of center sigma0, dispersion parameter theta and under the specified distance

Usage

dgmm(perm, sigma = identity.permutation(length(perm)), theta, dist.name = "kendall")

Arguments

perm permutation whose probability wants to be known
sigma0 central permutation of the GMM, by default the identity
theta vector dispersion parameter of the GMM
dist.name optional name of the distance used in the GMM. One of: kendall (default), cayley, hamming

Value

The probability of sigma in the given GMM

Examples

data <- matrix(c(1,2,3,4, 1,4,3,2, 1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
sig <- c(1,2,3,4)
theta <- c(0.1, 0.2, 0.3,1)
log.prob <- apply(data,MARGIN=1,FUN=function(x){log(dgmm(x,sig, theta, "hamming"))})
sum(log.prob)
dgmm (c(1,2,3,4), theta=c(1,1,1))
dgmm (c(1,2,3,4), theta=c(1,1,1), dist.name="cayley")
distance

Compute the distance between permutations

Description
Compute the distance between two given permutations. If only one permutation is given the other one is assumed to be the identity (1,2,3,....,n) The distance can be kendall, cayley, hamming and ulam

Usage
distance(perm1L permR \] identityNpermutation(length(perm1IIL distNname \] "kendall")

Arguments
perm1 a permutation
perm2 optional a permutation
dist.name optional. One of: kendall (default), cayley, hamming, ulam

Value
The distance between the permutations

Examples
distance(c(1LRLSLULTII
distance(c(1LRLSLULTIL c(1LRLSLULTII
distance(c(1LRLSLULTIL c(1LTLRLSLUIL "cayley")

dmm
Calculate the probability of a permutation in a MM

Description
Calculate the probability of a permutation sigma in a MM of center sigma0, dispersion parameter theta and under the specified distance

Usage
dmm(perm, sigma0 = identity.permutation(length(perm)), theta, dist.name = "kendall")
expectation.gmm

Arguments

- *perm*: permutation whose probability is asked for
- *sigma0*: optional central permutation of the MM, by default the identity
- *theta*: dispersion parameter of the MM
- *dist.name*: optional name of the distance used in the MM. One of: kendall (default), cayley, hamming, ulam

Value

The probability of sigma in the given MM

Examples

```r
data <- matrix(c(1,2,3, 4,1,4,3,2,1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
sig<--c(1,2,3,4)
log.prob <- apply(data,MARGIN=1,FUN=function(x){log(dmm(x,sig,1,"cayley"))})
sum(log.prob)
dmm(c(1,3,2,4), theta=0.1)
dmm(c(1,3,2,4), theta=0.1, dist.name="cayley")
dmm(c(1,3,2,4), theta=0.1, dist.name="hamming")
dmm(c(1,3,2,4), theta=0.1, dist.name="ulam")
```

expectation.gmm( theta, dist.name = "kendall")

Description

Compute the expected distance in the GMM under the Hamming distance

Usage

Arguments

- *theta*: n dimensional real vector with the dispersion parameters
- *dist.name*: optional name of the distance used in the GMM. One of: kendall (default), cayley, hamming

Value

The expected distance decomposition vector under the GMM

References

expectation.mm

Examples

```r
expectation.gmm(c(0.38, 0.44, 0.1, 0.2, 1, 0.1))
expectation.gmm(c(2, 2, 2, "cayley")
expectation.gmm(c(0.3, 0.1, 0.5, 0.1), "hamming")
```

---

**expectation.mm**

*Compute the expected distance, MM under the Hamming distance*

---

**Description**

Compute the expected distance in the MM under the Hamming distance

**Usage**

```r
expectation.mm(theta, perm.length, dist.name = "kendall")
```

**Arguments**

- `theta`: real dispersion parameter
- `perm.length`: length of the permutation in the considered model
- `dist.name`: optional name of the distance used in the MM. One of: kendall (default), cayley, hamming, ulam

**Value**

The expected distance under the MM

**References**


**Examples**

```r
expectation.mm(1, 7, "kendall")
expectation.mm(2, 5, "cayley")
expectation.mm(2, 4, "hamming")
expectation.mm(1, 6, "ulam")
```
freq.matrix  

Compute the frequency matrix

Description

Compute the first order marginal probability. In other words, given at least one permutation, calculate the proportion of them that have each item in each position.

Usage

freq.matrix(perm)

Arguments

perm  
a permutation or a collection of them

Value

A matrix with n rows and n columns with the proportion of the permutations in the input that have each item in each position.

Examples

freq.matrix(c(1,2,3,4,5))

generate.aux.files  

Generates the files for Ulam

Description

Generates files for Ulam which are aimed to accelerate the processes of counting the number of permutations at each distance, sampling and learning IFF these operations are going to be computed more than once.

Usage

generate.aux.files(perm.length)

Arguments

perm.length  
number of items in the permutations

Value

Nothing. Only writes in the current folder the auxiliary files.
**identity.permutation**

*Generate identity the permutation*

**Description**

This function generates the identity permutation of a given number of items.

**Usage**

```r
identity.permutation(perm.length)
```

**Arguments**

- `perm.length`: number of items in the permutation

**Value**

The identity permutation of the specified number of items.

**Examples**

```r
identity.permutation(3)
identity.permutation(7)
```

---

**insert**

*Insert operator*

**Description**

Given a permutation and two positions `i, j`, move item in position `i` to position `j`.

**Usage**

```r
insert(perm, i, j)
```

**Arguments**

- `perm`: a permutation
- `i`: position of the permutation
- `j`: position of the permutation
**Value**

The permutation in the input in which the operation has been applied

**Examples**

```r
insert(c(1LRLSLTLUILULRI, 5, 2))
insert(c(1LRLSLTLUILRLUI, 2, 5))
```

---

**inverse.perm**  
*Generate inverse permutation*

**Description**

This function generates the inverse of a given permutation. If the input is a matrix of permutations, invert all the permutations in the input.

**Usage**

```r
inverse.perm(perm)
```

**Arguments**

- `perm` a permutation or matrix of permutations

**Value**

The inverse permutation. If the input is a matrix, the matrix with the inverses

**Examples**

```r
inverse.perm(c(1, 2, 3, 4))
inverse.perm(c(2, 3, 4, 1))
data <- matrix(c(1, 2, 3, 4, 1, 4, 3, 2, 1, 2, 4, 3), nrow = 3, ncol = 4, byrow = TRUE)
inverse.perm(data)
```
inversion

inversion  Inversion operator

Description
Given a permutation and a position, swap positions $i$ and $i+1$

Usage
inversion(perm, i)

Arguments
<table>
<thead>
<tr>
<th>perm</th>
<th>a permutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>i</td>
<td>position of the permutation</td>
</tr>
</tbody>
</table>

Value
The permutation in the input with an inversion at the specified position

Examples
inversion(c(1,2,3,4,5),2)

is.permutation

is.permutation  Check if its argument is a permutation

Description
This function tests if the given argument is a permutation of the first n natural integers (excluding 0)

Usage
is.permutation(perm)

Arguments
| perm  | a vector (or a bidimensional matrix) |

Value
TRUE iff perm is a valid permutation (or a matrix of valid permutations)
Examples

```r
is.permutation(c(3,1,2,4))
is.permutation(c(6,1,2,3))
is.permutation(matrix(c(1,2,3, 4,1,4,3,2,1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE))
```

---

**lgmm**

Learn a Generalized Mallows Model

**Description**

Learn the parameter of the distribution of a sample of permutations coming from a Generalized Mallows Model (GMM).

**Usage**

```r
lgmm(data, sigma_0_ini = identity.permutation(dim(data)[2]),
     dist.name = "kendall", estimation = "approx")
```

**Arguments**

- `data`: the matrix with the permutations to estimate
- `sigma_0_ini`: optional the initial guess for the consensus permutation
- `dist.name`: optional name of the distance used by the GMM. One of: kendall (default), cayley, hamming
- `estimation`: optional select the approximated or the exact. One of: approx, exact

**Value**

A list with the parameters of the estimated distribution: the mode and the dispersion parameter vector

**References**


**Examples**

```r
data <- matrix(c(1,2,3,4, 1,4,3,2, 1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
lgmm(data, dist.name="kendall", estimation="approx")
lgmm(data, dist.name="cayley", estimation="approx")
lgmm(data, dist.name="cayley", estimation="exact")
lgmm(data, dist.name="hamming", estimation="approx")
```
**Description**

Compute the MLE for the dispersion parameter (theta) given a sample of n permutations and a central permutation.

**Usage**

```r
lgmm.theta(data, sigma_0 = identity.permutation(dim(data)[2]),
            dist.name = "kendall")
```

**Arguments**

- `data`: the matrix with the permutations to estimate.
- `sigma_0`: optional the initial guess for the consensus permutation. If not given it is assumed to be the identity permutation.
- `dist.name`: optional name of the distance used by the GMM. One of: kendall (default), cayley, hamming.

**Value**

The MLE for the dispersion parameter.

**Examples**

```r
data <- matrix(c(1,2,3,4, 1,4,3,2, 1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
lgmm.theta(data, dist.name="kendall")
lgmm.theta(data, dist.name="cayley")
lgmm.theta(data, dist.name="cayley", sigma_0=c(1,4,3,2))
lgmm.theta(data, dist.name="hamming")
```

---

**lmm**

*Learn a Mallows Model*

**Description**

Learn the parameter of the distribution of a sample of n permutations comming from a Mallows Model (MM).

**Usage**

```r
lmm(data, sigma_0_ini = identity.permutation(dim(data)[2]),
     dist.name = "kendall", estimation = "approx", disk = FALSE)
```
Arguments

- `data`: the matrix with the permutations to estimate
- `sigma_0_init`: optional the initial guess for the consensus permutation
- `dist.name`: optional the name of the distance used by the model. One of: kendall (default), cayley, hamming, ulam
- `estimation`: optional select the approximated or the exact. One of: approx, exact
- `disk`: optional can only be true if estimating a MM under the Ulam distance. Instead of generating the whole set of SYT and count of permutations per distance, it loads the info from a file in the disk

Value

A list with the parameters of the estimated distribution: the mode and the dispersion parameter

References


Examples

```r
data <- matrix(c(1,2,3,4, 1,4,3,2, 1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
lmm(data, dist.name="kendall", estimation="approx")
lmm(data, dist.name="cayley", estimation="approx")
lmm(data, dist.name="cayley", estimation="exact")
lmm(data, dist.name="hamming", estimation="exact")
lmm(data, dist.name="ulam", estimation="approx")
```

Description

Compute the MLE for the dispersion parameter (theta) given a sample of n permutations and a central permutation

Usage

```r
lmm.theta(data, sigma_0 = identity.permutation(dim(data)[2]),
          dist.name = "kendall", disk = FALSE)
```
marginal

Arguments

data the matrix with the permutations to estimate
sigma_0 optional the consensus permutation. If not given it is assumed to be the identity permutation
dist.name optional the name of the distance used by the model. One of: kendall (default), cayley, hamming, ulam
disk optional can only be true if estimating a MM under the Ulam distance. Instead of generating the whole set of SYT and count of permutations per distance, it loads the info from a file in the disk

Value

The MLE for the dispersion parameter

Examples

```r
data <- matrix(c(1,2,3,4, 1,4,3,2, 1,2,4,3), nrow = 3, ncol = 4, byrow = TRUE)
lmm.theta(data, dist.name="kendall")
lmm.theta(data, dist.name="cayley")
lmm.theta(data, dist.name="cayley", sigma_0=c(1,4,3,2))
lmm.theta(data, dist.name="hamming")
lmm.theta(data, dist.name="ulam")
```

marginal Compute the marginal probability, GMM under the Hamming distance

Description

Compute the marginal probability, GMM under the Hamming distance, of a distance decomposition vector for which some positions are known and some are not

Usage

```r
marginal(h, theta)
```

Arguments

h n dimensional distance decomposition vector where h_j = 0 means that $j$ is a fixed point, $h_j = 1$ means that $j$ is an unfixed point and otherwise $j$ is not known
theta n dimensional distance decomposition vector with the dispersion parameters

Value

The marginal probability
maxi.dist

References


Examples

marginal(c(1,0,1,NA,NA), c(0.1, 0.3, 0.7, 0.1, 1))
marginal(c(NA,0,1,NA,NA,0), c(0.1, 0.3, 0.7, 0.1, 0.7, 1))

maxi.dist  Get the maximum value of the distance between permutations

Description

Compute the maximum possible value for the distance between two given permutations. The distance can be kendall, cayley, hamming and ulam

Usage

maxi.dist(perm.length, dist.name = "kendall")

Arguments

perm.length number of items in the permutations
dist.name optional. One of: kendall (default), cayley, hamming, ulam

Value

The maximum value for the distance between the permutations

Examples

maxi.dist(4,"cayley")
maxi.dist(10,"ulam")
maxi.dist(4)
<table>
<thead>
<tr>
<th>order.ratings</th>
<th>Convert rating to permutation</th>
</tr>
</thead>
</table>

**Description**

This function is given a collection of ratings and converts each row to a permutation.

**Usage**

`order.ratings(ratings)`

**Arguments**

- `ratings`: a matrix in which each row is a vector of ratings of several items

**Value**

A matrix in which each row is the corresponding permutation of the items.

**Examples**

```r
data(ratings)
order.ratings(ratings)
```

<table>
<thead>
<tr>
<th>perm.sample.med</th>
<th>Sample of permutations</th>
</tr>
</thead>
</table>

**Description**

A R object containing a sample of permutations.

**Format**

Each row is a permutation.

<table>
<thead>
<tr>
<th>perm.sample.small</th>
<th>Sample of permutations</th>
</tr>
</thead>
</table>

**Description**

A R object containing a sample of permutations.

**Format**

Each row is a permutation.
perm2cycles

*Decompose a permutation in a set of cycles*

**Description**

Factor a given permutation in the set of independent cycles.

**Usage**

```r
perm2cycles(perm)
```

**Arguments**

- `perm` a permutation

**Value**

The permutation in the input in which the operation has been applied.

**Examples**

```r
perm2cycles(c(1,5,2,3,4))
```

perm2decomp

*Get the decomposition vector*

**Description**

Given a permutation and a distance name generate the decomposition vector.

**Usage**

```r
perm2decomp(perm, dist.name = "kendall")
```

**Arguments**

- `perm` the permutation
- `dist.name` optional the name of the distance. One of: kendall (default), cayley, hamming

**Value**

The distance decomposition vector of the given permutation and distance. For the Kendall distance is the inversion vector.
permutations.of

Examples
permutations.of(perm.length, alert = TRUE)

Arguments
perm.length  number of items in the permutation
alert        optional ask for confirmation when the number of permutations to show is very large

Value
A collection of every permutation of the specified number of items

Examples
permutations.of(3)
permutations.of(10)

rdist

Generate a collection of permutations at a given distance

Description
Given a number of permutations, the number of items in the permutations, a distance value and a distance name, generate a sample of permutations with the specified length at the given distance. Can be used to generate derangements and permutations of a given number of cycles

Usage
rdist(n, perm.length, dist.value, dist.name = "kendall")
Arguments

- \( n \) number of permutations in the sample
- \( \text{perm.length} \) number of items in the permutations
- \( \text{dist.value} \) distance value
- \( \text{dist.name} \) distance name. One of: kendall (default), cayley, hamming, ulam

Value

A sample of permutations at the given distance

Examples

```r
rdist(1L, TL R I
rdist(1L, TL RL "ulam")
len <- 3
rdist(n = 1L, perm.length = len, dist.value = len, "h") #derangement
cycles <- 2
rdist(n = 1L, perm.length = len, dist.value = len - cycles, "c") #permutation with 2 cycles
```

---

**read.perms**

Read a text file with a collection of permutations

Description

This function reads the text file in the specified path and checks if each row is a proper permutation

Usage

```r
read.perms(path)
```

Arguments

- \( \text{path} \) string with a path

Value

A collection of permutations in matrix form

Examples

```r
path = system.file("test.txt", package="PerMallows")
sample = read.perms(path)
```
Sample a Generalized Mallows Model

Description

Generate a sample of n permutations from a Generalized Mallows Model (GMM).

Usage

`rgmm(n, sigma0, theta, dist.name = "kendall", sampling.method = "multistage")`

Arguments

- `n` the number of permutations to be generated
- `sigma0` central permutation of the GMM
- `theta` dispersion parameter vector of the GMM
- `dist.name` optional used name of the distance used in the GMM. One of: kendall (default), cayley, hamming
- `sampling.method` optional name of the sampling algorithm. One of: multistage, gibbs (default)

Value

A matrix containing a sample of permutations from the specified distribution

References


Examples

`rgmm(2,c(1,2,3,4,5),c(1,1,1,1),"kendall", "multistage")`
`rgmm(2,c(1,2,3,4,5),c(1,1,1,1),"cayley", "multistage")`
`rgmm(2,c(1,2,3,4,5),c(1,1,1,1),"hamming", "multistage")`
`rgmm(2,c(1,2,3,4,5),c(1,1,1,1),"cayley", "gibbs")`
`rgmm(2,c(1,2,3,4,5),c(1,1,1,1),"hamming", "gibbs")`
Sample a Mallows Model

Description

Generate a sample of n permutations from a Mallows Model (MM).

Usage

\[
\text{rmm}(n, \sigma_0, \theta, \text{dist.name = "kendall"}, \text{sampling.method = NULL}, \\
\quad \text{disk = FALSE, alert = TRUE})
\]

Arguments

- \(n\): the number of permutations to be generated
- \(\sigma_0\): central permutation of the MM
- \(\theta\): dispersion parameter of the MM
- \(\text{dist.name}\): optional name of the distance used in the MM. One of: kendall (default), cayley, hamming, ulam
- \(\text{sampling.method}\): optional name of the sampling algorithm. One of: distances, multistage, gibbs (default)
- \(\text{disk}\): optional can only be true if using the Distances sampling algorithm for generating under the Ulam distance. Instead of generating the whole set of SYT and count of permutations per distance, it loads the info from a file in the disk
- \(\text{alert}\): check consistency of the parameters. TRUE by default

Value

A matrix containing a sample of permutations from the specified distribution

References


Examples

- `rmm(2, c(1,2,3,4,5), 1, "kendall", "distances")`
- `rmm(2, c(1,2,3,4,5), 1, "cayley", "distances")`
- `rmm(2, c(1,2,3,4,5), 1, "hamming", "distances")`
- `rmm(2, c(1,2,3,4,5), 1, "ulam", "distances")`
- `rmm(2, c(1,2,3,4,5), 1, "kendall", "multistage")`
- `rmm(2, c(1,2,3,4,5), 1, "cayley", "multistage")`
**runif.permutation**

**Random permutation**

**Description**

Generate a collection of n permutations uniformly at random

**Usage**

```r
runif.permutation(n = 1, perm.length)
```

**Arguments**

- `n` optional number of permutations to generate
- `perm.length` length of the permutations generated

**Value**

A single permutation or a matrix with n rows, each being a permutation. Every permutation is drawn uniformly at random and has length perm.length

**Examples**

```r
runif.permutation(1,5)
```

---

**swap**

**Swap two items of a permutation**

**Description**

Given a permutation and two position, swap both positions

**Usage**

```r
swap(perm, i, j)
```

**Arguments**

- `perm` a permutation
- `i` position of the permutation
- `j` position of the permutation

**Value**

The permutation in the input in which the two specified items have been swapped
Examples

swap(c(1, 2, 3, 4, 5), 2, 5)
Index

compose, 2
count.perms, 3
cycle2str, 4
cycles2perm, 4
data.apa, 5
data.order, 5
decomp2perm, 5
dgmm, 6
distance, 7
dmm, 7

expectation.gmm, 8
expectation.mm, 9

freq.matrix, 10
generate.aux.files, 10

identity.permutation, 11
insert, 11
inverse.perm, 12
inversion, 13
is.permutation, 13

lgmm, 14
lgmm.theta, 15
lmm, 15
lmm.theta, 16

marginal, 17
maxi.dist, 18

order.ratings, 19

perm.sample.med, 19
perm.sample.small, 19
perm2cycles, 20
perm2decomp, 20
permutations.of, 21

rdist, 21

read.perms, 22
rgmm, 23
rmm, 24
runif.permutation, 25

swap, 25