Package ‘expperm’

October 13, 2022

Type Package
Title Computing Expectations and Marginal Likelihoods for Permutations
Version 1.6
Date 2019-05-23
Author Ben Powell
Maintainer Ben Powell <ben.powell@york.ac.uk>
Description A set of functions for computing expected permutation matrices given a matrix of likelihoods for each individual assignment. It has been written to accompany the forthcoming paper ‘Computing expectations and marginal likelihoods for permutations’. Publication details will be updated as soon as they are finalized.
License GPL-3
Depends R (>= 2.10)
Imports Rcpp (>= 1.0.1)
LinkingTo Rcpp
LazyData true
RoxygenNote 6.1.1
Suggests testthat
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-05-28 21:03:06 UTC

R topics documented:

- expperm-package
- A
- BG
- BG_cpp
- brute
- brute_cpp
- df1
Description

A set of functions for computing expected permutation matrices given a matrix of likelihoods for each individual assignment. It has been written to accompany the forthcoming paper 'Computing expectations and marginal likelihoods for permutations'. Publication details will be updated as soon as they are finalized.

Details

The DESCRIPTION file:

Package: expperm
Type: Package
Title: Computing Expectations and Marginal Likelihoods for Permutations
Version: 1.6
Date: 2019-05-23
Author: Ben Powell
Maintainer: Ben Powell <ben.powell@york.ac.uk>
Description: A set of functions for computing expected permutation matrices given a matrix of likelihoods for each individual assignment.
License: GPL-3
Depends: R (>= 2.10)
Imports: Rcpp (>= 1.0.1)
LinkingTo: Rcpp
LazyData: true
RoxygenNote: 6.1.1
Suggests: testthat

Index of help topics:

A A small random matrix
BG The Brualdi-Gibson method for computing an expected permutation matrix
BG_cpp The Brualdi-Gibson method for computing an expected permutation matrix using C++
brute            Brute-force calculation of an expected permutation matrix
brute_cpp        Brute-force calculation of an expected permutation matrix using C++
df1              A small data frame of simulated records
df2              A (second) small data frame of simulated records
expperm-package  Computing Expectations and Marginal Likelihoods for Permutations
is.tridiagonal   Checking a matrix is tridiagonal
ryser            The Ryser method for computing an expected permutation matrix
ryser_cpp        The Ryser method for computing an expected permutation matrix using C++
sink             A variational approximation of an expected permutation matrix
sink_cpp         A variational approximation of an expected permutation matrix using C++
triA             A small random tridiagonal matrix

The package serves primarily to demonstrate the algorithms described in the accompanying paper, which is currently under review.

We include versions, which are as similar as reasonably possible, of algorithms written in both R and C++. The R code is intended to facilitate testing, modification and re-use of the code while the C++ code is intended to implement the algorithms most efficiently for application to real problems.

Author(s)
Ben Powell
Maintainer: Ben Powell <ben.powell@york.ac.uk>

References

A small random matrix

Description
A small random matrix used only to demonstrate the package’s algorithms in the examples sections of the package documentation.

Usage
A
**Format**

An object of class matrix with 7 rows and 7 columns.

---

**BG**  
*The Brualdi-Gibson method for computing an expected permutation matrix*

---

**Description**

Computes the expected permutation matrix and marginal likelihood from a tridiagonal matrix of assignment likelihoods using the Brualdi-Gibson method.

**Usage**

\[
\text{BG}(A, \text{return.permanent} = \text{FALSE})
\]

**Arguments**

- **A**: A tridiagonal matrix of assignment likelihoods.
- **return.permanent**: A logical value indicating whether the function should also return the permanent of \( A \), which is then added to the output as an attribute.

**Value**

\( E(P) \), the expected permutation matrix corresponding to \( A \).

**Examples**

```r
data(triA)
BG(triA)
```

---

**BG_cpp**  
*The Brualdi-Gibson method for computing an expected permutation matrix using C++*

---

**Description**

Computes the expected permutation matrix and marginal likelihood from a tridiagonal matrix of assignment likelihoods using the Brualdi-Gibson method.

**Usage**

\[
\text{BG\_cpp}(A)
\]
Arguments

A  A tridiagonal matrix of assignment likelihoods.

Value

\( E(P) \), the expected permutation matrix corresponding to \( A \).

Examples

data(triA)
BG_cpp(triA)

---

brute  Brute-force calculation of an expected permutation matrix

Description

Computes an expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods. The function literally enumerates all permutations so will be impractical for matrices with more than 10 rows.

Usage

brute(A, return.permanent = FALSE)

Arguments

A  A matrix of assignment likelihoods.

return.permanent  A logical value indicating whether the function should also return the permanent of \( A \), which is then added to the output as an attribute.

Value

\( E(P) \), the expected permutation matrix corresponding to \( A \).

Examples

data(A)
brute(A)
“Brute-force calculation of an expected permutation matrix using C++”

**Description**
Computes an expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods. The function literally enumerates all permutations so will be impractical for matrices with more than 10 rows.

**Usage**
```
brute_cpp(A)
```

**Arguments**
- `A`: A matrix of assignment likelihoods.

**Value**
- `E(P)`, the expected permutation matrix corresponding to `A`.

**Examples**
```
data(A)
brute_cpp(A)
```

---

**df1**

**Description**
A small data frame of simulated records as might be found in a population census. This data is used to demonstrate the package’s algorithms in a more realistic setting. It also allows for reproduction of the example towards the end of the paper that accompanies this package. The data is a subset of a larger set simulated by P. McLeod, R. Heasman and I. Forbes of the UK’s Office for National Statistics. At the time of publication this data is available at https://ec.europa.eu/eurostat/cros/content/job-training_en. The example below shows how we could compute a distance matrix for the records in dataframes df1 and df2.

**Usage**
```
df1
```

**Format**
An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 18 rows and 3 columns.
### Examples

```r
## Not run:
library(stringdist)
D <- matrix(, n, n)
for (i in 1:n) {
  for (j in 1:n) {
    D[i, j] <- stringdist(df1$PERNAME1[i], df2$PERNAME1[j]) +
    stringdist(df1$PERNAME2[i], df2$PERNAME2[j], method = "dl") +
    stringdist(df1$DOB_YEAR[i], df2$DOB_YEAR[j], method = "dl")
  }
}
## End(Not run)
```

---

### df2

_A (second) small data frame of simulated records_

### Description

A small data frame of simulated records as might be found in a population census. This data is used to demonstrate the package’s algorithms in a more realistic setting. It also allows for reproduction of the example towards the end of the paper that accompanies this package. The data is a subset of a larger set simulated by P. McLeod, R. Heasman and I. Forbes of the UK’s Office for National Statistics. At the time of publication this data is available at [https://ec.europa.eu/eurostat/cros/content/job-training_en](https://ec.europa.eu/eurostat/cros/content/job-training_en).

### Usage

`df2`

### Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 18 rows and 3 columns.

---

### is.tridiagonal

_Checking a matrix is tridiagonal_

### Description

A function for checking whether a matrix is tridiagonal. The check is used before attempting to apply the BG method for computing the permanent, since the method is only applicable to tridiagonal matrices.

### Usage

`is.tridiagonal(A)`
Arguments

A A matrix.

Value

A logical variable. TRUE if the A is tridiagonal, FALSE otherwise.

Examples

data(A)
is.tridiagonal(A)
data(triA)
is.tridiagonal(triA)

Description

Computes the expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods using the Ryser method.

Usage

ryser(A, return.permanent = FALSE)

Arguments

A A matrix of assignment likelihoods.
return.permanent A logical value indicating whether the function should also return the permanent of A, which is then added to the output as an attribute.

Value

E(P), the expected permutation matrix corresponding to A.

Examples

data(A)
ryser(A)
### ryser_cpp

The Ryser method for computing an expected permutation matrix using C++

---

**Description**

Computes the expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods using the Ryser algorithm.

**Usage**

```r
tyser_cpp(A)
```

**Arguments**

- **A**: A matrix of assignment likelihoods.

**Value**

- **E(P)**, the expected permutation matrix corresponding to A.

**Examples**

```r
data(A)
tyser_cpp(A)
```

---

### sink

A variational approximation of an expected permutation matrix

---

**Description**

Computes an approximate expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods. The approximation minimizes a constrained KL divergence from the likelihood, and is computed via the repeated renormalization of the input’s rows and columns.

**Usage**

```r
sink(A, maxit = 99, return.permanent.bound = FALSE)
```

**Arguments**

- **A**: A matrix of assignment likelihoods.
- **maxit**: An integer specifying the maximum number of steps used in the optimization.
- **return.permanent.bound**: A logical value indicating whether the function should also return an upper bound on the permanent of A, which is then added to the output as an attribute.
Value

\( E(P) \), the expected permutation matrix corresponding to \( A \).

Examples

data(A)
sink(A)

sink_cpp

A variational approximation of an expected permutation matrix using C++

Description

Computes an approximate expected permutation matrix and marginal likelihood from a matrix of assignment likelihoods. The approximation minimizes a constrained KL divergence from the likelihood, and is computed via the repeated renormalization of the input’s rows and columns.

Usage

sink_cpp(A, maxit = 99)

Arguments

A A matrix of assignment likelihoods.

maxit An integer specifying the maximum number of steps used in the optimization.

Value

\( E(P) \), the expected permutation matrix corresponding to \( A \).

Examples

data(A)
sink_cpp(A)
$\text{triA}$

<table>
<thead>
<tr>
<th>triA</th>
<th>A small random tridiagonal matrix</th>
</tr>
</thead>
</table>

**Description**

A small random tridiagonal matrix used only to demonstrate the package's algorithms in the examples sections of the package documentation.

**Usage**

triA

**Format**

An object of class `matrix` with 7 rows and 7 columns.
Index

* datasets
  A, 3
df1, 6
df2, 7
triA, 11
* linkage error
  expperm-package, 2
* math
  expperm-package, 2
* package
  expperm-package, 2
* permanent
  expperm-package, 2
* permutation
  expperm-package, 2

A, 3
BG, 4
BG_cpp, 4
brute, 5
brute_cpp, 6
df1, 6
df2, 7
expperm (expperm-package), 2
expperm-package, 2
is.tridiagonal, 7
ryser, 8
ryser_cpp, 9
sink, 9
sink_cpp, 10
triA, 11