

# Package ‘BosonSampling’

October 12, 2022

**Type** Package

**Title** Classical Boson Sampling

**Version** 0.1.3

**Date** 2021-11-07

**Description** Classical Boson Sampling using the algorithm of Clifford and Clifford (2017) <[arXiv:1706.01260](#)>. Also provides functions for generating random unitary matrices, evaluation of matrix permanents (both real and complex) and evaluation of complex permanent minors.

**Maintainer** Raphaël Clifford <[clifford@cs.bris.ac.uk](mailto:clifford@cs.bris.ac.uk)>

**License** GPL-2

**Imports** Rcpp (>= 0.12.12)

**LinkingTo** Rcpp, RcppArmadillo

**Encoding** latin1

**NeedsCompilation** yes

**Author** Peter Clifford [aut],  
Raphaël Clifford [cre, aut]

**Repository** CRAN

**Date/Publication** 2021-05-14 08:52:11 UTC

## R topics documented:

BosonSampling-package . . . . .	2
bosonSampler . . . . .	2
Permanent-functions . . . . .	4
randomUnitary . . . . .	5

<b>Index</b>	<b>6</b>
--------------	----------

---

BosonSampling-package *Classical Boson Sampling*

---

## Description

Classical Boson Sampling using the algorithm of Clifford and Clifford (2017) <arXiv:1706.01260>. Also provides functions for generating random unitary matrices, evaluation of matrix permanents (both real and complex) and evaluation of complex permanent minors.

## Details

Index of help topics:

BosonSampling-package	Classical Boson Sampling
Permanent-functions	Functions for evaluating matrix permanents
bosonSampler	Function for independently sampling from the Boson Sampling distribution
randomUnitary	Random unitary

## Author(s)

Peter Clifford <peter.clifford@jesus.ox.ac.uk> and Raphaël Clifford <clifford@cs.bris.ac.uk>

Maintainer: Raphaël Clifford <clifford@cs.bris.ac.uk>

---

bosonSampler	<i>Function for independently sampling from the Boson Sampling distribution</i>
--------------	---

---

## Description

The function implements the Boson Sampling algorithm defined in Clifford and Clifford (2017) <https://arxiv.org/abs/1706.01260>

## Usage

```
bosonSampler(A, sampleSize, perm = FALSE)
```

## Arguments

A	the first n columns of an (m x m) random unitary matrix, see <a href="#">randomUnitary</a>
sampleSize	the number of independent sample values required for given A
perm	TRUE if the permanents and pmfs of each sample value are required

## Details

Let the matrix  $A$  be the first  $n$  columns of an  $(m \times m)$  random unitary matrix, then  $X \leftarrow \text{bosonSampler}(A, \text{sampleSize} = N, \text{perm} = \text{TRUE})$  provides  $X\$values$ ,  $X\$perms$  and  $X\$pmfs$ ,

The component  $X\$values$  is an  $(n \times N)$  matrix with columns that are independent sample values from the Boson Sampling distribution. Each sample value is a vector of  $n$  integer-valued output modes in random order. The elements of the vector can be sorted in increasing order to provide a multiset representation of the sample value.

The outputs  $X\$perms$  and  $X\$pmfs$  are vectors of the permanents and probability mass functions (pmfs) associated with the sample values. The permanent associated with a sample value  $v = (v_1, \dots, v_n)$  is the permanent of an  $(n \times n)$  matrix constructed with rows  $v_1, \dots, v_n$  of  $A$ . Note the constructed matrix,  $M$ , may have repeated rows since  $v_1, \dots, v_n$  are not necessarily distinct. The pmf is calculated as  $\text{Mod}(pM)^2 / \text{prod}(\text{factorial}(\text{tabulate}(c)))$  where  $pM$  is the permanent of  $M$ .

## Value

$X = \text{bosonSampler}(A, \text{sampleSize} = N, \text{perm} = \text{TRUE})$  provides  $X\$values$ ,  $X\$perms$  and  $X\$pmfs$ . See Details.

## References

Clifford, P. and Clifford, R. (2017) The Classical Complexity of Boson Sampling, <https://arxiv.org/abs/1706.01260>

## Examples

```
set.seed(7)
n <- 20 # number of photons
m <- 200 # number of output modes
A <- randomUnitary(m)[,1:n]
# sample of output vectors
valueList <- bosonSampler(A, sampleSize = 10)$values
valueList
# sample of output multisets
apply(valueList,1, sort)
#
set.seed(7)
n <- 12 # number of photons
m <- 30 # number of output modes
A <- randomUnitary(m)[,1:n]
# sample of output vectors
valueList = bosonSampler(A, sampleSize = 1000)$values
# Compare frequency of output modes at different
# positions in the output vectors
matplot(1:m,apply(valueList,1,tabulate), pch =20, t = "p",
xlab = "output modes", ylab = "frequency")
```

**Description**

These three functions are used in the classical Boson Sampling problem

**Usage**

```
cxPerm(A)
rePerm(B)
cxPermMinors(C)
```

**Arguments**

A	a square complex matrix.
B	a square real matrix.
C	a rectangular complex matrix where $nrow(C) = ncol(C) + 1$ .

**Details**

Permanents are evaluated using Glynn's formula (equivalently that of Nijenhuis and Wilf (1978))

**Value**

cxPerm(A) returns a complex number: the permanent of the complex matrix A.  
rePerm(B) returns a real number: the permanent of the real matrix B.  
cxPermMinors(C) returns a complex vector of length  $ncol(C)+1$ : the permanents of all  $ncol(C)$ -dimensional square matrices constructed by removing individual rows from C.

**References**

Glynn, D.G. (2010) The permanent of a square matrix. *European Journal of Combinatorics*, **31**(7):1887–1891.  
Nijenhuis, A. and Wilf, H. S. (1978). *Combinatorial algorithms: for computers and calculators*. Academic press.

**Examples**

```
set.seed(7)
n <- 20
A <- randomUnitary(n)
cxPerm(A)
#
B <- Re(A)
rePerm(B)
#
C <- A[, -n]
```

```
v <- cxPermMinors(C)
#
# Check Laplace expansion by sub-permanents
c(cxPerm(A),sum(v*A[,n]))
```

---

randomUnitary	<i>Random unitary</i>
---------------	-----------------------

---

**Description**

Returns a square complex matrix sampled from the Haar random unitary distribution.

**Usage**

```
randomUnitary(size)
```

**Arguments**

size                    dimension of matrix

**Value**

A square complex matrix.

**Examples**

```
m <- 25 # size of matrix (m x m)
set.seed(7)
U <- randomUnitary(m)
#
n <- 5 # First n columns
A <- U[,1:n]
```

# Index

bosonSampler, [2](#)

BosonSampling-package, [2](#)

cxPerm (Permanent-functions), [4](#)

cxPermMinors (Permanent-functions), [4](#)

Permanent-functions, [4](#)

randomUnitary, [2](#), [5](#)

rePerm (Permanent-functions), [4](#)