Package ‘RaschSampler’

July 23, 2015

Type Package
Title Rasch Sampler
Version 0.8-8
Date 2015-07-23
Imports stats
Depends R (>= 3.0.0)
Description MCMC based sampling of binary matrices with fixed margins as used in exact Rasch model tests.
License GPL-2
NeedsCompilation yes
Author Patrick Mair [cre, aut],
    Reinhold Hatzinger [aut],
    Norman D. Verhelst [aut]
Maintainer Patrick Mair <mair@fas.harvard.edu>
Repository CRAN
Date/Publication 2015-07-23 15:13:53

R topics documented:

RaschSampler-package .... 2
phi.range .... 3
rsampler .... 4
RSctr .... 5
rsctrl .... 6
rsextrmat .... 7
rsextrobj .... 8
RSmpl .... 10
rstats .... 11
summary.RSctr .... 12
summary.RSmpl .... 13
xmpl .... 13

Index 15
Description

The package implements an MCMC algorithm for sampling of binary matrices with fixed margins complying to the Rasch model. Its stationary distribution is uniform. The algorithm also allows for square matrices with fixed diagonal.

Parameter estimates in the Rasch model only depend on the marginal totals of the data matrix that is used for the estimation. From this it follows that, if the model is valid, all binary matrices with the same marginals as the observed one are equally likely. For any statistic of the data matrix, one can approximate the null distribution, i.e., the distribution if the Rasch model is valid, by taking a random sample from the collection of equally likely data matrices and constructing the observed distribution of the statistic. One can then simply determine the exceedence probability of the statistic in the observed sample, and thus construct a non-parametric test of the Rasch model. The main purpose of this package is the implementation of a methodology to build nonparametric tests for the Rasch model.

In the context of social network theories, where the structure of binary asymmetric relations is studied, for example, person $a$ esteems person $b$, which corresponds to a 1 in cell $(a,b)$ of the associated adjacency matrix. If one wants to study the distribution of a statistic defined on the adjacency matrix and conditional on the marginal totals, one has to exclude the diagonal cells from consideration, i.e., by keeping the diagonal cells fixed at an arbitrary value. The RaschSampler package has implemented an appropriate option, thus it can be also used for sampling random adjacency matrices with given marginal totals.

Details

<table>
<thead>
<tr>
<th>Package:</th>
<th>RaschSampler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Package</td>
</tr>
<tr>
<td>Version:</td>
<td>0.8-8</td>
</tr>
<tr>
<td>Date:</td>
<td>2015-07-23</td>
</tr>
<tr>
<td>License:</td>
<td>GNU GPL 2, June 1991</td>
</tr>
</tbody>
</table>

The user has to supply a binary input matrix. After defining appropriate control parameters using `rsctrl` the sampling function `rsampler` may be called to obtain an object of class `RSmpl` which contains the generated random matrices in encoded form. After defining an appropriate function to operate on a binary matrix (e.g., calculate a statistic such as `phi.range`) the application of this function to the sampled matrices is performed using `rstats`. Prior to applying the user defined function, `rstats` decodes the matrices packed in the `RSmpl-object.

The package also defines a utility function `rsextrobj` for extracting certains parts from the `RSmpl-object resulting in an object of class `RSmplext`. Both types of objects can be saved and reloaded for
Summary methods are available to print information on these objects, as well as on the control object `R5ctr` which is obtained from using `rsctrl` containing the specification for the sampling routine.

**Note**

The current implementation allows for data matrices up to 4096 rows and 128 columns. This can be changed by setting `nmax` and `kmax` in `Raschsampler.f90` to values which are a power of 2. These values should also be changed in `rserror.R`.

For convenience, we reuse the Fortran code of package version 0.8-1 which circumvents the compiler bug in Linux distributions of GCC 4.3. In case of compilation errors (due to a bug in Linux distributions of GCC 4.3) please use `Raschsampler.f90` from package version 0.8-1 and change `nmax` and `kmax` accordingly (or use GCC 4.4).

**Author(s)**

Reinhold Hatzinger, Patrick Mair, Norman D. Verhelst

Maintainer: <mair@fas.harvard.edu>

**References**


---

**Example User Function**

**Description**

Calculates the $R_\phi$ statistic, i.e., the range of the inter-column correlations ($\phi$-coefficients) for a binary matrix.

**Usage**

`phi.range(mat)`

**Arguments**

- `mat` a binary matrix

**Value**

the range of the inter-column correlations
Examples

ctr <- rsctrl(burn_in = 10L, n_eff = 5L, step=10L, seed = 123L, tfixed = FALSE)
mat <- matrix(sample(c(0L,1L), 50L, replace = TRUE), nr = 10L)
rso <- rsampler(mat, ctr)
rso_st <- rstats(rso, phi.range)
print(unlist(rso_st))

rsampler

Sampling Binary Matrices

Description

The function implements an MCMC algorithm for sampling of binary matrices with fixed margins complying to the Rasch model. Its stationary distribution is uniform. The algorithm also allows for square matrices with fixed diagonal.

Usage

rsampler(inpmat, controls = rsctrl())

Arguments

inpmat  A binary (data) matrix with \( n \) rows and \( k \) columns.
controls  An object of class RSctr. If not specified, the default parameters as returned by function rsctrl are used.

Details

rsampler is a wrapper function for a Fortran routine to generate binary random matrices based on an input matrix. On output the generated binary matrices are integer encoded. For further processing of the generated matrices use the function rstats.

Value

A list of class RSmpl with components

n  number of rows of the input matrix
k  number of columns of the input matrix
inpmat  the input matrix
tfixed  TRUE, if diagonals of inpmat are fixed
burn_in  length of the burn in process
n_eff  number of generated matrices (effective matrices)
step  controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note in rsctrl).
seed  starting value for the random number generator
n_tot  number of matrices in outvec, \( n_{\text{tot}} = n_{\text{eff}} + 1 \)
outvec  vector of encoded random matrices
ier  error code
Note
An element of `outvec` is a four byte (or 32 bits) integer. The matrices to be output are stored bitwise (some bits are unused, since a integer is used for every row of a matrix. So the number of integers per row needed equals \((k + 31)/32\) (integer division), which is one to four in the present implementation since the number of columns and rows must not exceed 128 and 4096, respectively.

The summary method (\texttt{summary.RSmp1}) prints information on the content of the output object.

Author(s)
Reinhold Hatzinger, Norman Verhelst

References

See Also
\texttt{rsctrl}, \texttt{rstats}

Examples
```r
data(xmpl)
ctr<-rsctrl(burn_in=10, n_eff=5, step=10, seed=0, tfixed=FALSE)
res<-rsampler(xmpl,ctr)
summary(res)
```
### Description

Various parameters that control aspects of the random generation of binary matrices.

### Usage

```r
rsctrl(burn_in = 100, n_eff = 100, step = 16, seed = 0, tfixed = FALSE)
```

### Arguments

- **burn_in**: the number of sampled matrices to come close to a stationary distribution. The default is `burn_in = 100`. (The actual number is `2 * burn_in * step`.)
- **n_eff**: the number of effective matrices, i.e., the number of matrices to be generated by the sampling function `rsampler`. `n_eff` must be positive and not larger than 8191 (2^13-1). The default is `n_eff = 100`.
- **step**: controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note below). The default is `step = 16`.
- **seed**: is the indicator for the seed of the random number generator. Its value must be in the range 0 and 2147483646 (2^31-2). If the value of seed equals zero, a seed is generated by the sampling function `rsampler` (dependent on the system’s clock) and its value is returned in the output. If seed is not equal to zero, its value is used as the seed of the random number generator. In that case its value is unaltered at output. The default is `seed = 0`.
- **tfixed**: logical, – specifies if in case of a quadratic input matrix the diagonal is considered fixed (see note below). The default is `tfixed = FALSE`.
**Value**

A list of class RSctr with components burn_in, n_eff, step, seed, tfixed.

**Note**

If one of the components is incorrectly specified the error function rserror is called and some informations are printed. The output object will not be defined.

The specification of step controls the sampling algorithm as follows: If, e.g., burn_in = 10, n_eff = 5, and step = 2, then during the burn in period step * burn_in = 2 * 10 matrices are generated. After that, n_eff * step = 5 * 2 matrices are generated and every second matrix of these last ten is returned from link{rsampler}.

tfixed has no effect if the input matrix is not quadratic, i.e., all matrix elements are considered free (unrestricted). If the input matrix is quadratic, and tfixed = TRUE, the main diagonal of the matrix is considered as fixed. On return from link{rsampler} all diagonal elements of the generated matrices are set to zero. This specification applies, e.g., to analyzing square incidence matrices representing binary asymmetric relation in social network theory.

The summary method (summary.RSctr) prints the current definitions.

**See Also**

rsampler

**Examples**

```r
ctr <- rsctrl(n_eff = 1L, seed = 987654321)  # specify new controls
summary(ctr)

## Not run:
ctr2 <- rsctrl(step = -3, n_eff = 10000)  # incorrect specifications

## End(Not run)
```

---

**rsextrmat**

**Extracting a Matrix**

**Description**

Convenience function to extract a matrix.

**Usage**

```r
rsextrmat(RSobj, mat.no = 1)
```
Arguments

RSobj object as obtained from using rsampler or rsextrobj
mat.no number of the matrix to extract from the sample object.

Value

One of the matrices (either the original or a sampled matrix)

See Also

rsampler, rsextrobj, rstats.

Examples

ctr <- rsctrl(burn_in = 10, n_eff = 3, step = 10, seed = 0, tfixed = FALSE)
mat <- matrix(sample(c(0, 1), 50, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
summary(all_m)

# extract the third sampled matrix (here the fourth)
third_m <- rsextmat(all_m, 4)
head(third_m)

rsextrobj

Extracting Encoded Sample Matrices

Description

Utility function to extract some of the generated matrices, still in encoded form.

Usage

rsextrobj(RSobj, start = 1, end = 8192)

Arguments

RSobj object as obtained from using rsampler
start number of the matrix to start with. When specifying 1 (the default value) the original input matrix is included in the output object.
end last matrix to be extracted. If end is not specified, all matrices from RSobj are extracted (the maximal value is 8192, see rsctrl). If end is larger than the number of matrices stored in RSobj, end is set to the highest possible value (i.e., n_tot).
**Value**

A list of class `RSmpl` with components

- `n`: number of rows of the input matrix
- `k`: number of columns of the input matrix
- `inpmat`: the input matrix
- `tfixed`: TRUE, if diagonals of `inpmat` are fixed
- `burn_in`: length of the burn in process
- `n_eff`: number of generated matrices (effective matrices)
- `step`: controls the number of void matrices generated in the burn in process and when effective matrices are generated (see note in `rsctrl`).
- `seed`: starting value for the random number generator
- `n_tot`: number of matrices in `outvec`
- `outvec`: vector of encoded random matrices
- `ier`: error code

**Note**

By default, all generated matrices plus the original matrix (in position 1) are contained in `outvec`, thus `n_tot = n_eff + 1`. If the original matrix is not in `outvec` then `n_tot = n_eff`.

For saving and loading objects of class `RSobj` see the example below.

For extracting a decoded (directly usable) matrix use `rsextrmat`.

**See Also**

`rsampler`, `rsextrmat`

**Examples**

```r
ctr <- rsctrl(burn_in = 10L, n_eff = 3L, step = 10L, seed = 0L, tfixed = FALSE)
mat <- matrix(sample(c(0L, 1L), 50L, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
some_m <- rsextrobject(all_m, 1L, 2L)
summary(all_m)
summary(some_m)

## Not run:
save(some_m, file = "some.RSobj")
some_new <- load("some.RSobj")
summary(some_new)

## End(Not run)
```


Sample Objects

Description

The objects of class RSmp1 and RSmp1ext contain the original input matrix, the generated (encoded) random matrices, and some information about the sampling process.

Value

A list of class RSmp1 or RSmp1ext with components

- `n`: number of rows of the input matrix
- `k`: number of columns of the input matrix
- `inpmat`: the input matrix
- `tfixed`: TRUE, if diagonals of inpmat are fixed
- `burn_in`: length of the burn in process
- `n_eff`: number of generated matrices (effective matrices)
- `step`: controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note in `rsctrl`).
- `seed`: starting value for the random number generator
- `n_tot`: number of matrices in outvec.
- `outvec`: vector of encoded random matrices
- `ier`: error code (see below)

Generation

These classes of objects are returned from `rsampler` and `rsextrobj`.

Methods

Both classes have methods for the generic `summary` function.

Note

By default, all generated matrices plus the original matrix (in position 1) are contained in outvec, thus `n_tot = n_eff + 1`. If the original matrix is not in outvec then `n_tot = n_eff`.

If `ier` is 0, no error was detected. Otherwise use the error function `rserror(ier)` to obtain some informations.

For saving and loading objects of class RSmp1 or RSmp1ext see the example in `rsextrobj`.

See Also

`rsampler, rsextrobj`
Calculating Statistics for the Sampled Matrices

Description

This function is used to calculate user defined statistics for the (original and) sampled matrices. A user defined function has to be provided.

Usage

```
rstats(RSobj, userfunc, ...)```

Arguments

- `RSobj`: object as obtained from using `rsampler` or `rsextrobj`
- `userfunc`: a user defined function which performs operations on the (original and) sampled matrices. The first argument in the definition of the user function must be an object of type matrix.
- `...`: further arguments, that are passed to the user function

Value

A list of objects as specified in the user supplied function

Note

The encoded matrices that are contained in the input object `RSobj` are decoded and passed to the user function in turn. If `RSobj` is not an object obtained from either `rsampler` or `rsextrobj` or no user function is specified an error message is printed. A simple user function, `phi.range`, is included in the RaschSampler package for demonstration purposes.

`rstats` can be used to obtain the 0/1 values for any of the sampled matrices (see second example below). Please note, that the output from the user function is stored in a list where the number of components corresponds to the number of matrices passed to the user function (see third example).

See Also

- `rsampler`, `rsextrobj`

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 5, step=10, seed = 12345678, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
rso <- rsampler(mat, ctr)
rso_st <- rstats(rso, phi.range)
unlist(rso_st)
```
# extract the third generated matrix
# (here, the first is the input matrix)
# and decode it into rsmat

rso2 <- rsextrobj(rso, 4, 4)
summary(rso2)

rsmat <- rstats(rso2, function(x) matrix(x, nr = rso2$n))
print(rsmat[[1]])

# extract only the first r rows of the third generated matrix

mat <- function(x, nr = nr, r = 3){
  m <- matrix(x, nr = nr)
  m[1:r,]
}

rsmat2 <- rstats(rso2, mat, nr = rso$n, r = 3)
print(rsmat2[[1]])

# apply a user function to the decoded object

print(phi.range(rsmat[[1]]))

summary.RSctr

Summary Method for Control Objects

Description

Prints the current definitions for the sampling function.

Usage

## S3 method for class 'RSctr'
summary(object, ...)

Arguments

object
  object of class RSctr as obtained from rsctrl

...  
  potential further arguments (ignored)

See Also

rsctrl

Examples

ctr <- rsctrl(n_eff = 1, seed = 123123123)  # specify controls
summary ctr

summary(RSctr)
Summary Methods for Sample Objects

Description
Prints a summary list for sample objects of class `RSmpl` and `RSmpext`.

Usage
```r
## S3 method for class 'RSmpl'
summary(object, ...)
## S3 method for class 'RSmpext'
summary(object, ...)
```

Arguments
- `object`: object as obtained from `rsampler` or `rsextrobj`
- `...`: potential further arguments (ignored)

Details
Describes the status of an sample object.

See Also
`rsampler`, `rsextrobj`

Examples
```r
ctr <- rsctrl(burn_in = 10, n_eff = 3, step = 10, seed = 0, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
summary(all_m)

some_m <- rsextrobj(all_m, 1, 2)
summary(some_m)
```

Example Data

Description
Fictitious data sets - matrices with binary responses

Usage
`data(xmpl)`
Format

The format of `xmpl` is:
300 rows (referring to subjects)
30 columns (referring to items)

The format of `xmplbig` is:
4096 rows (referring to subjects)
128 columns (referring to items)
`xmplbig` has the maximum dimensions that the RaschSampler package can handle currently.

Examples

```r
data(xmpl)
print(head(xmpl))
```
Index

*Topic datasets
  xmpl, 13
*Topic misc
  phi.range, 3
  rsampler, 4
  RScstr, 5
  rsctrl, 6
  rsextrmat, 7
  rsextrobj, 8
  RSmpl, 10
  rstats, 11
  summary.RScttr, 12
  summary.RSmpl, 13
*Topic package
  RaschSampler-package, 2

phi.range, 2, 3, 11
RaschSampler (RaschSampler-package), 2
RaschSampler-package, 2
rsampler, 2, 4, 5–11, 13
RScstr, 3–5, 5
rsctrl, 2–6, 6, 9, 10, 12
rsextrmat, 7, 9
rsextrobj, 2, 8, 8, 10, 11, 13
RSmpl, 2, 4, 9, 10, 13
RSmplext, 2, 13
RSmplext (RSmpl), 10
rstats, 2, 4, 5, 8, 11
summary.RScttr, 7, 12
summary.RSmpl, 5, 13
summary.RSmplext (summary.RSmpl), 13

xmpl, 13
xmplbig (xmpl), 13