# The Rasch Sampler

Verhelst, Norman D.; Hatzinger, Reinhold; Mair, Patrick

*Published in:*
Journal of Statistical Software

*DOI:*
10.18637/jss.v020.i04

Published: 01/01/2007

*Document Version*
Publisher's PDF, also known as Version of record

*Journal of Statistical Software*

*May 2007, Volume 20, Issue 4.* http://www.jstatsoft.org/

# The Rasch Sampler

**Norman D. Verhelst**
National Institute for
Educational Measurement (Cito)

**Reinhold Hatzinger**
Wirtschaftsuniversität Wien

**Patrick Mair**
Wirtschaftsuniversität Wien

**Abstract**

The Rasch sampler is an efficient algorithm to sample binary matrices with given marginal sums. It is a Markov chain Monte Carlo (MCMC) algorithm. The program can handle matrices of up to 1024 rows and 64 columns. A special option allows to sample square matrices with given marginals and fixed main diagonal, a problem prominent in social network analysis. In all cases the stationary distribution is uniform. The user has control on the serial dependency.

*Keywords*: Markov chain Monte Carlo, binary matrices, fixed marginals, nonparametric tests, Rasch model.

## 1. Introduction

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 (Besag and Clifford 1989). One can then simply determine the exceedance probability of the statistic in the observed sample (its $p$-value), and thus construct a non-parametric test of the Rasch model. The accuracy of the observed (i.e., simulated) distribution will depend on the sample size, and this in turn will only depend on the time one can or wants to spend to draw the random sample.

The MCMC methodology for this problem has been used by a number of authors (Connor and Simberloff 1979; Besag and Clifford 1989; Roberts and Stone 1990; Guttorp 1995; Rao, Jana, and Bandyopadhyay 1996; Ponocny 2001). They all use the same basic approach, henceforth referred to as the classical approach, which can be briefly characterized as follows:

1. All binary matrices with given marginals, the sample space, are considered as states in a finite Markov chain.
2. From any state, the matrix $A_t$ say, the process can move to any state of a subset of the sample space. This subset will be called the neighborhood of the matrix $A_t$. This will be described in more detail in the next section.
   (a) The neighborhood is relatively small compared to the size of the sample space.
   (b) The actual choice for moving from the current state to the next is by random sampling from the neighborhood.
3. The stationary distribution is not uniform: the probability of each matrix is proportional to the size of its neighborhood. Rao *et al.* (1996) offer some heuristics to approach the uniform distribution.

The algorithm that is implemented in the Rasch sampler differs in three important ways from the classical MCMC algorithms:

1. The neighborhoods of the states are much larger than in the classical approach.
2. The choice of the next state is based on importance sampling, not on random sampling from the neighborhood.
3. The stationary distribution is uniform by an application of the Metropolis-Hastings algorithm to the basic transition matrix.

These three issues will be discussed in three separate sections. In a separate section the algorithm will be extended to the case of square matrices with fixed diagonals. Proofs of propositions will be omitted since these are published elsewhere (Verhelst 2007).

## 2. Tetrad and binomial neighborhoods

Let $\Sigma_{\mathbf{rc}}$ be the sample space, i.e., the set of all binary matrices with row totals given by the $n$-dimensional vector $\mathbf{r}$, and column totals given by the $k$-dimensional vector $\mathbf{c}$. To exclude the possibility that $\Sigma_{\mathbf{rc}} = \emptyset$, it will be assumed that there exists a matrix with these marginals. Usually this will be an observed matrix, that will be denoted as $A_0$.

### 2.1. The case of $n \times 2$ matrices

The concepts of tetrad and binomial neighborhoods will be introduced by considering the special case of matrices with two columns. Since in such matrices each row consists of one out of four possible patterns, such a matrix, $A$ say, can be summarized like in Table 1.

| pattern | | frequency |
|---|---|---|
| 0 | 0 | $u$ |
| 1 | 1 | $v$ |
| 1 | 0 | $a$ |
| 0 | 1 | $b$ |

Table 1: Summary of a $n \times 2$ matrix.

A tetrad is defined as a $2 \times 2$ submatrix of $A$ (by selecting 2 rows) that has one of the following forms:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

A tetrad transformation consists of changing either form of (1) into the other for a single tetrad. By such a transformation the row and column totals of the matrix do not change. The tetrad neighborhood of matrix $A$ is the set of all matrices that can be obtained by a tetrad transformation on $A$. This set will be denoted by $\mathcal{A}_T(A)$. Notice that $A$ itself does not belong to its tetrad neighborhood. From Table 1, it is clear that

$$\#\mathcal{A}_T(A) = a \times b. \tag{2}$$

By a finite sequence of tetrad transformations all matrices belonging to $\Sigma_{\mathbf{rc}}$ can be reached.

In the matrix represented by Table 1, there are $m = a + b$ rows with a row total of one. A binomial operation consists of assigning a one to the first position and a zero to the second position for $a$ of these $m$ rows. The $b$ other rows have the complementary pattern. By such an operation on $A$, the marginals do not change. A binomial transform of $A$ is a matrix that results from a binomial operation on $A$ and that is not equal to $A$. The binomial neighborhood of $A$ is defined as the set of binomial transforms of $A$, and denoted by $\mathcal{A}_B(A)$. From Table 1 it will be clear that

$$\#\mathcal{A}_B(A) = \binom{a+b}{a} - 1 = \binom{m}{a} - 1, \tag{3}$$

and that

$$\mathcal{A}_T(A) \subset \mathcal{A}_B(A) \text{ if } a, b > 1, \tag{4}$$

i.e., a tetrad transform is a binomial transform.

## 2.2. The general case of $n \times k$ matrices.

For each of the $\binom{k}{2}$ column pairs of a matrix $A \in \Sigma_{\mathbf{rc}}$, a submatrix consisting of these two columns can be formed, and a table like Table 1 can be constructed. Applying a binomial transformation on this submatrix, and putting the transformed columns back in the original matrix will be called a binomial transformation of the original matrix. The notation is adapted as follows. Column pairs are denoted by the ordered pair $(i, j)$, $(i < j \leq k)$. The number of rows in the submatrix having a row total of one is denoted as $m_{ij}(A)$, the number of these rows having a one in column $i$ is denoted as $a_{ij}(A)$ and $b_{ij}(A) = m_{ij}(A) - a_{ij}(A)$. The explicit reference to the matrix $A$ will be dropped if it is clear from the context which matrix is referred to. The set of matrices that can be constructed by applying a binomial transformation on the column pair $(i, j)$ will be called the $B_{ij}$ neighborhood of $A$ and will be denoted as $\mathcal{A}_B^{(i,j)}(A)$.

A similar notation can be applied for tetrad neighborhoods: the $T_{ij}$ neighborhood of $A$ is the set of matrices that can be formed by a tetrad transformation on the column pair $(i, j)$ of $A$, and is denoted as $\mathcal{A}_T^{(i,j)}(A)$. Next, $B$- and $T$-neighborhoods are defined.

**Definition 1.** *The binomial neighborhood (or $B$-neighborhood) of matrix $A \in \Sigma_{\mathbf{rc}}$ is*

$$\mathcal{A}_B(A) = \bigcup_{(i,j)} \mathcal{A}_B^{(i,j)}(A).$$

**Definition 2.** *The tetrad neighborhood (or $T$-neighborhood) of matrix $A \in \Sigma_{\mathbf{rc}}$ is*

$$\mathcal{A}_T(A) = \bigcup_{(i,j)} \mathcal{A}_T^{(i,j)}(A).$$

An easy to prove but important result is given in Verhelst (2007).

**Proposition 1** *For any two column pairs $(i, j)$ and $(g, h)$ of $A \in \Sigma_{\mathbf{rc}}$ with $(i, j) \neq (g, h)$, it holds that $\mathcal{A}_B^{(i,j)}(A) \cap \mathcal{A}_B^{(g,h)}(A) = \emptyset$.*

A similar result holds for tetrad neighborhoods:

**Corollary 1.** *For any two column pairs $(i, j)$ and $(g, h)$ of $A \in \Sigma_{\mathbf{rc}}$ with $(i, j) \neq (g, h)$, it holds that $\mathcal{A}_T^{(i,j)}(A) \cap \mathcal{A}_T^{(g,h)}(A) = \emptyset$.*

It may be the case that for a column pair $(i, j)$ it holds that $a_{ij} = 0$ or $b_{ij} = 0$, in which case there are no tetrads for this pair.

**Definition 3.** *The column pair $(i, j)$ is a Guttman pair if $a_{ij} \times b_{ij} = 0$. If $a_{ij} \times b_{ij} > 0$, the pair will be called regular.*

**Definition 4.** *The $k_2$-measure of $A \in \Sigma_{\mathbf{rc}}$ is defined as $k_2(A) = \#\{(i, j) : i < j \leq k,\ (i, j)$ is a regular pair.*

In Verhelst (2007), a simple but useful proposition is proved:

**Proposition 2.** *If $k_2(A) = 0$ for some $A \in \Sigma_{\mathbf{rc}}$, then $\#\Sigma_{\mathbf{rc}} = 1$ and $\Sigma_{\mathbf{rc}} = \{A\}$.*

From (3), Proposition 1, and Definition 4, it is easily deduced that

$$\#\mathcal{A}_B(A) = \sum_{i<j} \#\mathcal{A}_B^{(i,j)}(A) = \sum_{a_{ij} \times b_{ij} > 0} \binom{m_{ij}}{a_{ij}} - k_2(A). \tag{5}$$

and using (2) and Corollary 1 gives

$$\#\mathcal{A}_T(A) = \sum_{i<j} \#\mathcal{A}_T^{(i,j)}(A) = \sum_{i<j} a_{ij} \times b_{ij} \tag{6}$$

| | Tetrad | Binomial | estimate of $\#\Sigma_{\mathbf{rc}}$ |
|---|---|---|---|
| Case 1 | $1.23 \times 10^6$ | $7.91 \times 10^{40}$ | $7.83 \times 10^{1774}$ |
| Case 2 | $1.12 \times 10^6$ | $6.00 \times 10^{40}$ | $2.28 \times 10^{1709}$ |

Table 2: Sizes of neighborhoods.

In the testing phase of the program, two artificial data matrices with 300 rows and 30 columns were created. In Table 2, the sizes of their tetrad and binomial neighborhoods are displayed, together with an estimate of the sizes of the two sample spaces. For details, see Verhelst (2007); for the estimate of the size of the sample space, see also Chen, Diaconis, Holmes, and Liu (2005).

## 3. The basic transition matrix of the Markov chain

The transition matrix of a Markov chain, $P = (p_{st})$, contains the transition probabilities, where $p_{st}$ is the probability that the next state is $s$ given that the current state is state $t$. In the classical approach, $p_{st}$ is defined as

$$p_{st} = \begin{cases} [\#\mathcal{A}_T(A_t)]^{-1} & \text{if } A_s \in \mathcal{A}_T(A_t), \\ \\ 0 & \text{otherwise.} \end{cases}$$

representing simple random sampling from the tetrad neighborhood. It is well know that this Markov chain is irreducible, and therefore has a stationary distribution (Rao *et al.* 1996). Using binomial neighborhoods, a similar sampling scheme can be followed, by sampling randomly from the binomial neighborhood of the current matrix. It turns out, however, that this sampling scheme is not very efficient. The variability of $\#\mathcal{A}_B^{(i,j)}(A_t)$ over column pairs can be considerable, with the consequence that a binomial transformation is applied to the same column pair for very long sequences of transitions. To avoid this, a transition matrix $Q = (q_{st})$ for the Markov chain is defined as

$$q_{st} = w_{st} \times d_t, \tag{7}$$

where

$$d_t = [k_2(A_t)]^{-1} \tag{8}$$

and

$$w_{st} = \begin{cases} \left[\binom{m_{ij}}{a_{ij}} - 1\right]^{-1} & \text{if } A_s \in \mathcal{A}_B^{(i,j)}(A_t), \\ \\ 0 \text{ if } A_s \notin \mathcal{A}_B(A_t). \end{cases} \tag{9}$$

Notice that $d_t$ in (8) is not well defined only in case $k_2(A_t) = 0$, but then, from Proposition 2, $\#\Sigma_{\mathbf{rc}} = 1$, and the problem is solved. This trivial case will be left out of consideration. Notice also that (9) is well defined and unambiguous because of Proposition 1: if $A_s \in \mathcal{A}_B(A_t)$, then there exists exactly one pair of columns, $(i, j)$ say, such that $A_s \in \mathcal{A}_B^{(i,j)}(A_t)$. The stationary distribution of the Markov chain defined by (7) is given in the Proposition 3. The proof is given in Verhelst (2007).

**Proposition 3.** *$Q\pi = \pi$ with $\pi_t \propto k_2(A_t)$, $(t = 1, \ldots, \#\Sigma_{\mathbf{rc}})$.*

The Markov chain can be implemented, using the following algorithm.

**Algorithm 1. Importance sampling from the binomial neighborhood of $A$**

1. *Select randomly a pair of columns from the $k_2(A)$ regular column pairs of $A$.*

2. *Apply a random binomial operation to the selected pair. If the resulting matrix equals $A$, repeat step 2; otherwise the resulting matrix is the new state.*

Notice that all matrices from $\mathcal{A}_B(A)$ have a positive probability of being sampled, but the probabilities are not equal, because the probability that column pair $(i, j)$ is sampled is $[k_2(A)]^{-1}$ if $\mathcal{A}_B^{(i,j)}(A) \notin \emptyset$, but otherwise independent of $\#\mathcal{A}_B^{(i,j)}(A)$.

## 4. The Metropolis-Hastings algorithm

Extensive experimentation with Algorithm 1 showed that the variability in the $k_2$-measures is often very small but not zero, so that the stationary distribution of $Q$ is not uniform. The Metropolis-Hastings algorithm is an elegant procedure to make the stationary distribution uniform. In general, the algorithm transforms a defined transition matrix, $Q$ say, into a transition matrix $Q^*$ with a prespecified vector $\pi$ as stationary vector. The algorithm is defined as (see, e.g., Tanner (1994) for a concise description)

$$q_{st}^* = \alpha_{st} \times q_{st}, \ (s \neq t), \tag{10}$$

where

$$\alpha_{st} = \begin{cases} \min\left[\dfrac{\pi_s q_{ts}}{\pi_t q_{st}}, 1\right] & \text{if } \pi_t q_{st} > 0, \\ 1 & \text{if } \pi_t q_{st} = 0. \end{cases} \tag{11}$$

The diagonal elements are given by

$$q_{tt}^* = 1 - \sum_{s \neq t} q_{st}^*.$$

If we want the uniform stationary distribution, then $\pi_s/\pi_t = 1$. Taking $Q$ as defined by (7), because $w_{st} = w_{ts}$ (Verhelst 2007), we obtain that

$$\frac{q_{ts}}{q_{st}} = \frac{k_2(A_t)}{k_2(A_s)},$$

whence we can easily deduce our final result:

**Algorithm 2. Importance sampling and Metropolis-Hastings**

1. *Select randomly a pair of columns from the* $k_2(A)$ *regular column pairs of* $A$.
2. *Apply a random binomial operation to the selected pair, yielding* $A^*$.
   (a) *If* $A^* = A$, *repeat step 2.*
   (b) *Otherwise,*
      i. *If* $k_2(A^*) \leq k_2(A)$, *the new state is* $A^*$,
      ii. *If* $k_2(A^*) > k_2(A)$, *then the new state remains* $A$ *with probability* $1-k_2(A)/k_2(A^*)$, *otherwise the new state is* $A^*$.

## 5. Square matrices with fixed diagonals

In social network theory, often a square incidence matrix, representing a binary asymmetric relation has to be analyzed. Having a tool to sample from the set of all binary matrices with given marginals and with the main diagonal values kept at their (arbitrary) value is valuable for social network research (Wasserman 1977). In this section we discuss an adaptation of Algorithm 2 to fit these cases.

The adaptation is relatively simple, thanks to a nice theorem of Rao *et al.* (1996). They start from a simple observation. In a $3 \times 3$ subtable of the matrix, formed by taking out three rows and the corresponding columns, the following two patterns, called alternating hexagons, can occur:

$$\begin{pmatrix} * & 1 & 0 \\ 0 & * & 1 \\ 1 & 0 & * \end{pmatrix} \text{ or } \begin{pmatrix} * & 0 & 1 \\ 1 & * & 0 \\ 0 & 1 & * \end{pmatrix}, \tag{12}$$

where $*$ indicates a fixed value. Notice that in neither of the two tables there is a tetrad, and yet, the marginals of the table do not change when an alternating hexagon is replaced by its complement, i.e., by a hexagon transformation. But this implies that in general not all matrices can be constructed by a finite sequence of tetrad transformations.

In their Theorem 2, Rao *et al.* (1996) prove that in case of a square binary matrix with fixed marginals and a fixed main diagonal, all matrices can be constructed from any other matrix by a finite sequence of tetrad transformations and hexagon operations.

Now it is easy to extend Algorithm 2 to apply also to this case. We do this by a number of easy to check features of the neighborhood:

1. The definition of the binomial neighborhood has to be adapted a little bit in a straightforward manner, in the sense that $m_{ij}$ is the number of rows in column pair $(i, j)$ that are of the form $(0, 1)$ or $(1, 0)$. Patterns containing a fixed value are not counted but are treated in the same way as the patterns $(0, 0)$ or $(1, 1)$.
2. If we apply a single hexagon operation to matrix $A$, exactly three columns of the matrix are changed, so that the resulting matrix cannot belong to the tetrad or binomial neighborhood of $A$.

3. So in a natural way we can define the hexagon neighborhood of a matrix $A$ as the set of matrices that can be constructed with a single hexagon transformation. This neighborhood will denoted as $\mathcal{A}_H(A)$.

4. Since a hexagon operation changes three columns of the matrix, and a tetrad or a binomial transformation change two columns, it is clear that

$$\mathcal{A}_H(A) \cap \mathcal{A}_T(A) = \mathcal{A}_H(A) \cap \mathcal{A}_B(A) = \emptyset,$$

so that the combined tetrad-hexagon or binomial-hexagon neighborhoods are partitioned much in the same way as the tetrad or binomial neighborhoods.

5. Next we define for all $A \in \Sigma_{\mathbf{rc}}$, the $k_3$-measure as

$$k_3(A) = \begin{cases} k_2(A) \text{ if } \#\mathcal{A}_H(A) = 0, \\ \\ k_2(A) + 1 \text{ otherwise.} \end{cases} \tag{13}$$

Without giving all the details, we can describe an algorithm which is very closely related to Algorithm 2.

**Algorithm 3. Importance sampling from the combined binomial-hexagon neighborhood of $A$**

1. *If $k_2(A) = k_3(A)$, apply Algorithm 2.*

2. *If $k_2(A) < k_3(A)$, draw a random number $u$ uniformly from $(0, 1]$;*

   (a) *if $u \leq [k_3(A)]^{-1}$, apply the hexagon operation to a randomly drawn alternating hexagon from the $\#\ A_H(A)$ alternating hexagons in $A$, yielding matrix $A^*$. If $k_3(A) < k_3(A^*)$, select $A$ as the next matrix with probability $1 - k_3(A)/k_3(A^*)$, else select $A^*$.*

   (b) *else, apply Algorithm 2.*

# 6. An implementation in R: The RaschSampler package

## 6.1. General structure

The workhorse of the **RaschSampler** is a FORTRAN 95 subroutine which implements the MCMC algorithm described in this paper as an R (R Development Core Team 2007) package. It is called from the R (wrapper) function `rsampler`. Apart from several parameters controlling the algorithm (to be explained in Section 6.2) it expects a binary matrix (usually the reponses from $n$ subjects to $k$ items, or a $k \times k$ adjacency matrix) as input and returns the generated matrices in encoded form. The output is stored in a list together with all control parameters to allow further operations (calculating statistics, saving the results of the sampler, replicating the sampling process, etc.). To make use of these sampled matrices the user has to define an appropriate R function that operates on each of the generated matrices,

e.g., calculating a statistic (for convenience the **RaschSampler** package contains an example user function `phi` which calculates the range of all inter-column correlations, denoted as $R_\varphi$ henceforth). The second main function of the package is `rstats`, a (wrapper) function which decodes the sampled matrices and passes them to the user defined function in turn. The output of `rstats` is a list with the same number of elements as the number of matrices passed to it. Each element contains the output from the user defined function, e.g., the statistic(s) for a (sampled) matrix.

All other functions in the package are utility functions, for defining the control parameters, printing the status of the output lists, and extracting some of the generated matrices.

### 6.2. User control

The use of MCMC methodology has two serious drawbacks. The first one is that the stationary distribution is an asymptotic distribution, and one can never be sure to sample from this distribution after a finite number of steps through the sample space. The second drawback has to do with serial dependency: even when the stationary distribution has been reached, a sequence of draws from the sample space will show autocorrelations, meaning that the sampled matrices are not independent of each other, even if their distribution is uniform.

To face these two problems, two practical tools are provided in the program, by means of the two tuning parameters `burn_in` and `step`. These are explained in turn.

If $Q$ is a transition matrix of a Markov chain with a uniform stationary distribution, then $Q^\ell,\ \ell = 1, 2, \ldots$also has the same stationary vector. In practice, this means that $\ell$ steps governed by $Q$ will be carried out before the resulting matrix is considered as a sampled matrix. As $\ell$ increases, the serial dependency will decrease. In the program the value of $\ell$ is at the discretion of the user by the tuning parameter `step`.

To allow the process to approximate its stationary distribution, it is customary to use a number of idle steps through the sample space, i.e. sampling matrices that are not used to compute the statistic(s) of interest. This number is under the control of the user by the tuning parameter `burn_in`. Notice, however, that the number of matrices generated before the real sampling starts equals the product of the two parameters `burn_in` and `step`.

After the burn in period, a number of matrices are sampled, using $Q^\ell$. This number is the sample size and is governed by the tuning parameter `n_eff`.

If the program is run, the total number of matrices generated equals

```
step * (burn_in + n_eff)
```

However, on output the resulting list contains only `n_eff + 1` (encoded) matrices, i.e., `n_eff` sampled plus the original input matrix (in position 1).

The other user controls concern the `seed` of the random number generator (the default value of zero generates a random seed based on the system time, the used seed value is returned from the sampler) and the parameter `tfixed` that enables the sampling of square matrices with a fixed diagonal (e.g., adjacency matrices).

### 6.3. Some examples

For the first example we use a (ficitious) data matrix `xmpl` with $n = 300$ rows and $k = 30$ columns provided in the **RaschSampler** package.

```
R> data("xmpl")
```

The default control parameters can be obtained by calling the control function without any arguments.

```
R> ctr <- rsctrl()
R> summary(ctr)

Current sampler control specifications in ctr :
    burn_in =  100
    n_eff =  100
    step =  16
    seed =  0
    tfixed =  FALSE
```

We want to generate 5 random matrices and use a random starting value for the random number generator. The control object is redefined by

```
R> ctr <- rsctrl(n_eff = 5, seed = 0)
R> summary(ctr)

Current sampler control specifications in ctr :
    burn_in =  100
    n_eff =  5
    step =  16
    seed =  0
    tfixed =  FALSE
```

We call the sampler, store the results in `result1` and print summary information.

```
R> result1 <- rsampler(xmpl, ctr)
R> summary(result1)

Status of object result1 after call to RSampler:
    n  =  300
    k =  30
    burn_in =  100
    n_eff =  5
    step =  16
    seed =  690790426
    tfixed =  FALSE
    n_tot =  6
    outvec contains 1800 elements
```

`n_tot` specifies the number of encoded matrices stored in `result1`, `outvec` contains these matrices, the first is always the original input matrix. The matrices are stored row-wise as integers (one integer is needed for $k \leq 32$, two integers for $33 < k \leq 64$, for further details see

the package help files). Since $n = 300$, $k < 32$, and five matrices have been sampled `outvec` has `300 * (5 + 1)` elements. The seed has been set to 690790426.

To calculate statistics for these matrices we use `rstats` with the sampling result and the (predefined) user function `phi` as arguments and store the results in `stat1`:

```
R> stat1 <- rstats(result1, phi)
```

The output is

```
R> unlist(stat1)

        1         2         3         4         5         6
0.3517041 0.3101368 0.3615032 0.4026465 0.3178987 0.3102170
```

As mentioned in Section 6.1 the output from `rstats` returns the values (statistics) from the user defined function specified as the second argument in the call of `rstats()`. For this example, we can see that the range of all item intercorrelations for the (fictitious) data matrix, i.e. $R_\varphi = 0.3517041$, is of similar magnitude as the corresponding values for the five sampled matrices (stored in positions 2 – 6).

A utility function `rsextrobj` has been defined to allow for extracting certain parts from the sampling result. For example, if we want to perform the statistics function only to the first two generated matrices we might use

```
R> resultparts <- rsextrobj(result1, start = 2, end = 3)
R> summary(resultparts)

Status of extracted object resultparts :
    n  =  300
    k =  30
    burn_in =  100
    n_eff =  2
    step =  16
    seed =  123
    tfixed =  FALSE
    n_tot =  2
    outvec contains 600 elements

R> stat1parts <- rstats(resultparts, phi)
R> unlist(stat1parts)

        1         2
0.3661533 0.3874665
```

The summary shows that `resultparts` is an extracted object and that the number of contained matrices is `n_tot = 2`. Since `n_eff`, the number of sampled matrices is also 2 we know that the original matrix is no longer contained in the list `resultparts`.

Original sampling results as well as extracted parts can be saved and reloaded for later usage, e.g., by

```
R> save(stat1parts, file = "some.RSobj")
R> again <- load("some.RSobj")
```

The second example examines the quantile for the statistic $R_\varphi$ (as implemented in the example user function `phi`) for a larger data matrix. The data matrix `xmplbig` is provided in in the **RaschSampler** package and has $n = 1024$ rows and $k = 64$ comlumns.

```
R> data("xmplbig")
```

The sampling process is specified to generate `n_eff = 1000` matrices and the starting value for the random number generator is fixed to `seed = 987654321`. The other control parameters are set to their default values.

```
R> ctr <- rsctrl(n_eff = 1000, seed = 987654321)
R> summary(ctr)

Current sampler control specifications in ctr :
    burn_in =  100
    n_eff =  1000
    step =  16
    seed =  987654321
    tfixed =  FALSE
```

Generating the 1000 random matrices by

```
R> result2 <- rsampler(xmplbig, ctr)
```

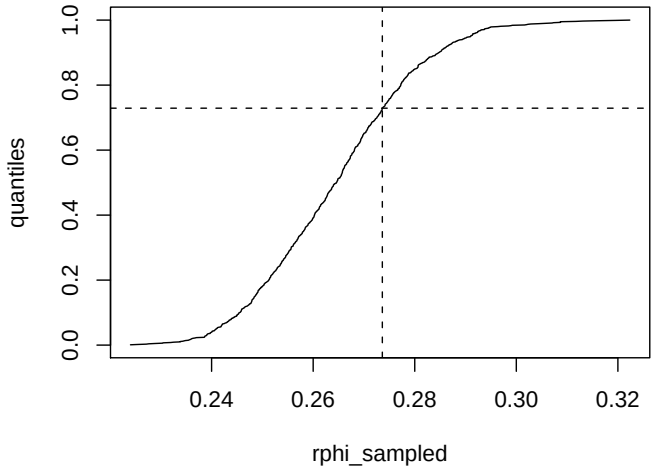


Figure 1: A quantile plot of the $R_\varphi$ statistics for the generated matrices, `n_eff = 1000`. The dashed lines indicate the position of the $R_\varphi$ statistic for the input matrix.

took 45 seconds on an 1.6GHz Intel(R) Pentium(R) M processor.
After applying the statistics function

```
R> stat2 <- rstats(result2, phi)
```

the results might be displayed graphically as shown in Figure 1 using some R commands such as

```
R> stat2 <- unlist(stat2)
R> rphi_obs <- stat2[1]
R> rphi_sampled <- sort(stat2[-1])
R> quantiles <- seq(0.001, 1, by = 0.001)
R> plot(rphi_sampled, quantiles, type = "l")
R> ycoord <- length(rphi_sampled[rphi_sampled <= rphi_obs])/1000
R> abline(h = ycoord, lty = 2)
R> abline(v = rphi_obs, lty = 2)
```

## References


Besag J, Clifford P (1989). "Generalized Monte Carlo Significance Tests." *Biometrika*, **76**, 633–642.

Chen Y, Diaconis P, Holmes S, Liu J (2005). "Sequential Monte Carlo Methods for Statistical Analysis of Tables." *Journal of the American Statistical Association*, **100**, 109–120.

Connor E, Simberloff D (1979). "The Assembly of Species Communities: Chance or Competition." *Ecology*, **60**, 1132–1140.

Guttorp P (1995). *Stochastic Modeling of Scientific Data*. Chapman and Hall, London.

Ponocny I (2001). "Nonparametric Goodness-Of-Fit Tests for the Rasch Model." *Psychometrika*, **66**, 437–460.

Rao A, Jana R, Bandyopadhyay S (1996). "A Markov Chain Monte Carlo Method for Generating Random $(0, 1)$-Matrices with Given Marginals." *Sankhya A*, **58**, 225–242.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Roberts A, Stone L (1990). "Island Sharing by Archipelago Species." *Oecologia*, **83**, 560–567.

Tanner M (1994). *Tools for Statistical Inference*. Springer-Verlag, New York.

Verhelst N (2007). "An Efficient MCMC-Algorithm to Sample Binary Matrices With Fixed Marginals." *Psychometrika*. In press.

Wasserman S (1977). "Random Directed Graph Distributions and the Triad Census in Social Networks." *Journal of Mathematical Sociology*, **5**, 61–86.

**Affiliation:**

Norman D. Verhelst
National Institute for Educational Measurement (Cito)
Arnhem, The Netherlands
E-mail: norman.verhelst@cito.nl
URL: http://www.cito.nl

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 20, Issue 4 *Submitted:* 2006-10-01
May 2007 *Accepted:* 2007-02-22