The pcm2sampler - Part II

How to use the `pcm2sampler`

Kathrin Gruber
Institute for Service Marketing and Tourism, WU Wien
Norman Verhelst
Eurometrics, Tiel, The Netherlands

Overview

- The `pcm2sampler`
- Exact tests
- How to use the `pcm2sampler`

General structure

- workhorse is a FORTRAN 95 subroutine `samplerPCM2`
- main programm is written in `R` (wrapper) and is called from the function `pcm2sampler`

- **Input:** a matrix consisting of binary and\or ternary items (entries), several parameters for controlling the algorithm

- **Output:** list of generated matrices (and control parameters)

- **Further operations:** calculate statistics (exact tests), replicate the sampling process, saving the results, ...

Input

```
pcm2sampler(inpmat, controls = ctrl())
```

- **inpmat**: the input matrix (binary and\or ternary entries), with n = number of rows (subjects, maximum number 1023) and k = number of columns (items, maximum number 63)

- **controls**: parameters for controlling the algorithm specified by the function `ctrl()`

```
ctrl(burn_in, n_eff, step, t_fixed, seed)
```

Tuning parameters (1)

Approximation of the stationary distribution

- **burn_in**: the number of burn-in blocks (≥ 0)
  to start the process somewhere near the stationary distribution

Control over serial dependency

- **step**: stepsize (> 0)
  to lower the serial dependency between the outcomes
  (add.: to influence the extend of the steps of the process within the sample space)

  controls the number of void matrices in the burn in process and when effective matrices are generated

Burn-in period = burn_in $\times$ step

Tuning parameters (2)

- **n_eff**: number of sampled matrices after the burn-in period (the sample size)
  maximum number of effective matrices is 10,000

E.g. step = 5, burn_in = 200, 200×5 = 1000 matrices are generated before the first effective matrix

$$\boxed{\text{Total number of generated matrices} = \text{step} \times (\text{burn\_in} + \text{n\_eff})}$$

No. of void matrices between two effective matrices = step - 1

Tuning parameters (3)

- **seed**: seed of the random number generator
  $= 0$: seed is generated by the subroutine and the value is stored (on output) in the parameter seed
  $\neq 0$: seed is used as the seed in the random number generator and on output it has the same value as on input

- **t_fixed**: logical, must be false upon calling (not implemented yet)

Output

After defining appropriate control parameters using `ctrl()` the sampling function `pcm2sampler()` is called to obtain an object which contains the generated random matrices in encoded form.

- **outvec**: contains `n_eff + 1` encoded matrices (sampled plus the original input matrix in position 1)
  Matrices are stored column-wise, with each column starting in a new element of outvec

- **n_tot**: number of encoded matrices

Additional methods

- **summary()**: generic function, method to control and sample objects
- **extrmat()**: function for extracting a matrix
- **extrobj()**: function for extracting encoded sample matrices

Example (1)

```
> ctr <- ctrl()
> summary(ctr)

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

Example (2)

```
> data(xmpl)
> ctr<-ctrl(burn_in=10, n_eff=5, step=10, seed=0, t_fixed=FALSE)
> res<-pcm2sampler(xmpl,ctr)
> summary(res)

Status of object res after call to pcm2ampler:
        n = 300
        k = 30
        burn_in = 10
        n_eff = 5
        step = 10
        seed = 115940628
        t_fixed = FALSE
        n_tot = 6
        outvec contains 1800 elements
```

Short introduction to exact tests (1)

Statistical tests and confidence intervals are based on exact probability statements that are valid for any sample size.

Motivation for exact tests:

- no parameter estimation needed
- do not base on asymptotic and approximative statistical methods
- also valid for small sample size

Short introduction to exact tests (2)

Construction principle in general:

- Rearrange the labels of the observed data points.
- Calculate all possible values of the test statistic (to derive the test statistic under H_0 is valid)

Short introduction to exact tests (3)

More specific:

- Sample all possible matrices from Σ_{rs} with identical margins r and s
- Calculate test statistic $T(A_0)$ for the observed data matrix A_0
- Calculate $T(A_1) \ldots T(A_n)$ for the simulated data matrices to derive the nonparametric distribution of T
- Evaluate exceedance probability of $T(A_0)$ by counting the number of $T(A_j) \geq T(A_0)$ for $(j = 1, \ldots, n)$
- Reject H_0 if

$$\left(p = \frac{1}{n} \sum \mathbb{I}_{\{T(A_j) \geq T(A_0)\}} \right) \leq \alpha$$

How to derive a test statistics using the `pcm2sampler?` (1)

Define an appropriate `R` function that operates on each of the generated matrices by the use of the function `rstats()`

Example 1:

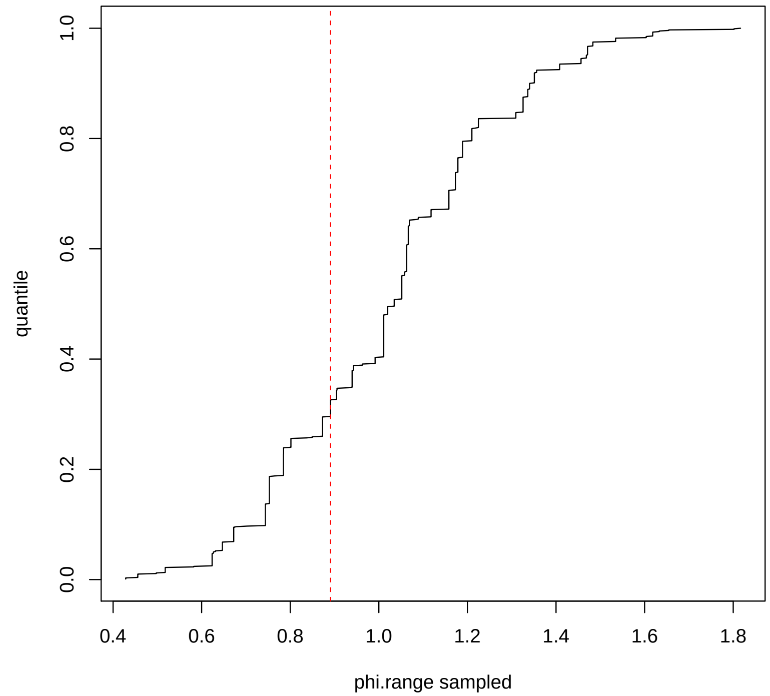
Calculates the R_ϕ statistic (the range of the inter-column correlations (ϕ-coefficients) for a binary matrix)

```
> ctr <- ctrl(burn_in = 10, n_eff = 5, step=10, seed = 123, t_fixed = FALSE)
> mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
> rso <- pcm2sampler(mat, ctr)
> rso_st <- rstats(rso,phi.range)
> print(unlist(rso_st))

        1         2         3         4         5         6
0.8908708 1.3403061 1.5345225 1.0517837 1.0629020 1.3093073
```

How to derive a test statistics using the `pcm2sampler`? (2)

Generating 1000 random matrices



How to derive a test statistics using the `pcm2sampler?` (3)

Example 2:

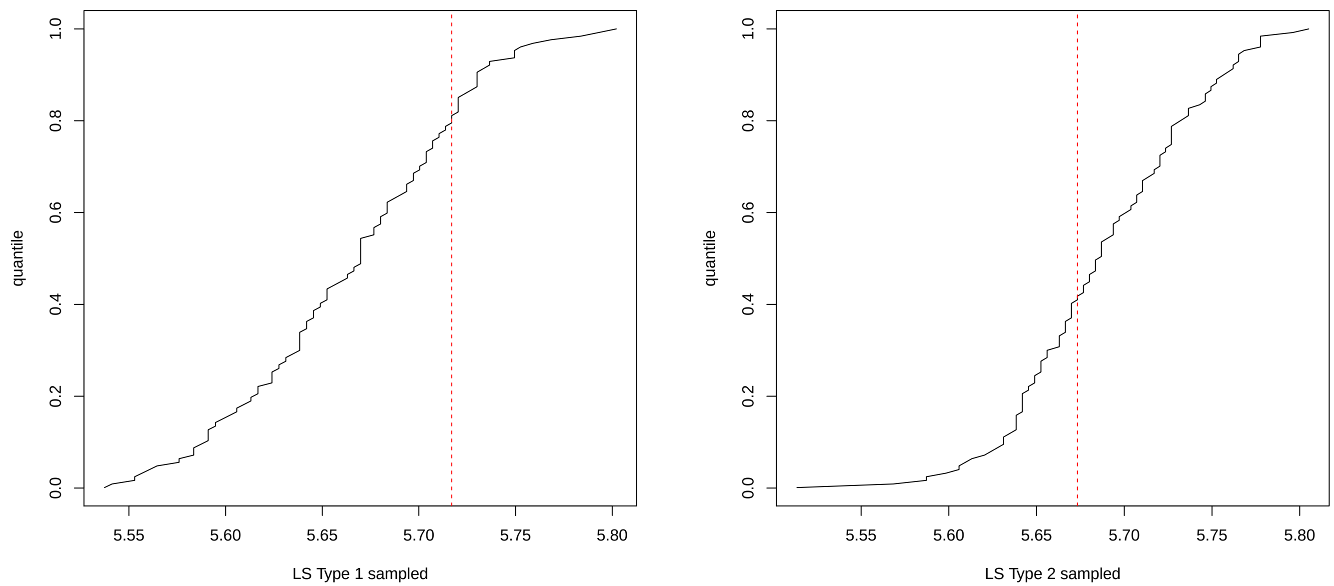
Calculate a statistic that is operating on the number of Latin Squares of type I or type II

$$\log(1 + \sharp LS1) \text{ or } \log(1 + \sharp LS2)$$

matrix of size n = 10, k = 100
all 10 items are ternary!

```
> NumLS1
[1] 303
> NumLS2
[1] 290
```

How to derive a test statistics using the `pcm2sampler`? (4)




```
> LS1[1]
[1] 5.717028
> LS2[1]
[1] 5.673323
```

Y. Chen and D. Small. Exact Tests for the Rasch Model via Sequential Importance Sampling. *Psychometrika, 70(1):11–30, 2005.*

G.N. Masters. A Rasch Model for Partial Credit Scoring. Psychometrika, 47(2):149–174, 1982.

I. Ponocny. Nonparametric goodness-of fit tests for the Rasch model. *Psychometrika, 66:437–460, 2001.*

A. Rao, R. Jana and S. Bandyopadhyay. A Makrov Chain Monte Carlo Method for Generating Random (0,1)-Matrices with Given Marginals. Sankhya A, 58: 225-242, 1196.

T. Snijders. Enumeration and Simulation Methods for 0-1 Matrices with Given Marginals. *Psychometrika, 56(3):397–417, 1991.*

N.D. Verhelst. An efficient MCMC algorithm to sample binary matrices with fixed marginals. Psychometrika, 73:705–728, 2008.

N.D. Verhelst, R. Hatzinger and M. Mair. The Rasch Sampler. *Journal of Statistical Software, 20(4), 2007*