Package 'qrng'

April 20, 2023

Version 0.0-9

Encoding UTF-8

Title (Randomized) Quasi-Random Number Generators

Description Functionality for generating (randomized) quasi-random numbers in high dimensions.

Author Marius Hofert [aut, cre],
Christiane Lemieux [aut]

Maintainer Marius Hofert <mhofert@hku.hk>

Depends R (>= 3.0.0)

Imports utils

Suggests randtoolbox, copula

Enhances

License GPL-2 | GPL-3

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-04-20 12:40:02 UTC

Repository/R-Forge/Project qrng

Repository/R-Forge/Revision 81

Repository/R-Forge/DateTimeStamp 2023-04-19 20:16:28

R topics documented:

qrng *Compute Quasi-Random Sequences*

Description

Computing Korobov, generalize Halton and Sobol' quasi-random sequences.

Usage

```
korobov(n, d = 1, generator, randomize = c("none", "shift"))
ghalton(n, d = 1, method = c("generalized", "halton"))
sobol  (n, d = 1, randomize = c("none", "digital.shift", "Owen", "Faure.Tezuka",
                                "Owen.Faure.Tezuka"), seed, skip = 0, ...)
```

Arguments

n	number n of points to be generated ≥ 2.
d	dimension d.
generator	numeric of length d or length 1 (in which case it is appropriately extended to length d). All numbers must be in $\{1, \ldots, n\}$ and must be (coercible to) integers.
randomize	character string indicating the type of randomization for the point set. korobov() one of the following: **"none"** no randomization. **"shift"** a uniform random variate modulo 1. sobol() one of the following: "none" no randomization. **"digital.shift"** a digital shift. **"Owen", "Faure.Tezuka","Owen.Faure.Tezuka"** calls sobol() from package **randtoolbox** with scrambling being 1, 2 and 3, respectively. If randomize is a logical, then it is interpreted as "none" if FALSE and "digital.shift" if TRUE.
method	character string indicating which sequence is generated, generalized Halton or (plain) Halton.
seed	if provided, an integer used within set.seed() for the non-scrambling randomize methods (so "none" or "digital.shift") or passed to the underlying sobol() from package **randtoolbox** for the scrambling methods. If not provided, the non-scrambling methods respect a global set.seed() but the scrambling methods do not (they then use 4711 as default); see sobol() from package **randtoolbox** for details.
skip	number of initial points in the sequence to be skipped (skip = 0 means the sequence starts with at the origin). Note that for the scrambling methods this simply computes n + skip points and omits the first skip-many.
...	additional arguments passed to sobol() from package **randtoolbox**.

Details

For `sobol()` examples see `demo(sobol_examples)`. In particular, be careful when using `skip > 0` and `randomize = TRUE`; in this case, choosing a wrong seed (or no seed) might lead to a bad sequence.

Note that these procedures call fast C code. The following restrictions apply:

korobov() `n,d` must be $\leq 2^{31} - 1$.

ghalton() `n` must be $\leq 2^{32} - 1$ and `d` must be ≤ 360.

sobol() if `randomize = "none"` or `randomize = "digital.shift"`, `n` must be $\leq 2^{31} - 1$ and `d` must be ≤ 16510.

The choice of parameters for `korobov()` is crucial for the quality of this quasi-random sequence (only basic sanity checks are conducted). For more details, see l'Ecuyer and Lemieux (2000).

The generalized Halton sequence uses the scrambling factors of Faure and Lemieux (2009).

Value

`korobov()` and `ghalton()` return an (n, d)-`matrix`; for $d = 1$ an n-vector is returned.

Author(s)

Marius Hofert and Christiane Lemieux

References

Faure, H., Lemieux, C. (2009). Generalized Halton Sequences in 2008: A Comparative Study. *ACM-TOMACS* **19**(4), Article 15.

l'Ecuyer, P., Lemieux, C. (2000). Variance Reduction via Lattice Rules. *Stochastic Models and Simulation*, 1214–1235.

Lemieux, C., Cieslak, M., Luttmer, K. (2004). RandQMC User's guide. See https://www.math.uwaterloo.ca/~clemieux/randq

Examples

```
n <- 1021 # prime
d <- 4 # dimension

## Korobov's sequence
generator <- 76 # see l'Ecuyer and Lemieux
u <- korobov(n, d = d, generator = generator)
pairs(u, gap = 0, pch = ".", labels = as.expression(
      sapply(1:d, function(j) bquote(italic(u[.(j)])))))

## Randomized Korobov's sequence
set.seed(271)
u <- korobov(n, d = d, generator = generator, randomize = "shift")
pairs(u, gap = 0, pch = ".", labels = as.expression(
      sapply(1:d, function(j) bquote(italic(u[.(j)])))))

## Generalized Halton sequence (randomized by definition)
```

```
set.seed(271)
u <- ghalton(n, d)
pairs(u, gap = 0, pch = ".", labels = as.expression(
      sapply(1:d, function(j) bquote(italic(u[.(j)])))))

## For sobol() examples, see demo(sobol_examples)
```

`test_functions` *Test Functions*

Description

Functions for testing low-discrepancy sequences.

Usage

```
sum_of_squares(u)
sobol_g(u, copula = copula::indepCopula(dim = ncol(u)), alpha = 1:ncol(u), ...)
exceedance(x, q, p = 0.99, method = c("indicator", "individual.given.sum.exceeds",
                                      "sum.given.sum.exceeds"))
```

Arguments

`u`	(n, d)-matrix containing n d-dimensional realizations (of a potential quasi-random number generator). For `sum_of_squares()` these need to be marginally standard uniform and for `sobol_g()` they need to follow the copula specified by `copula`.
`copula`	`Copula` object for which the inverse Rosenblatt transformation exists.
`alpha`	vector of parameters of Sobol's g test function.
`...`	additional arguments passed to the underlying `cCopula()`.
`x`	(n, d)-matrix containing n d-dimensional realizations.
`q`	**"indicator"** d-vector containing the componentwise thresholds; if a number it is recycled to a d-vector. **"individual.given.sum.exceeds", "sum.given.sum.exceeds"** threshold for the sum (row sums of `x`).
`p`	If `q` is not provided, the probability `p` is used to determine `q`. **"indicator"** d-vector containing the probabilities determining componentwise thresholds via empirical quantiles; if a number, it is recycled to a d-vector. **"individual.given.sum.exceeds", "sum.given.sum.exceeds"** probability determining the threshold for the sum (row sums of `x`) via the corresponding empirical quantile.
`method`	`character` string indicating the type of exceedance computed (see Section Value below).

Details

For examples see the demo `man_test_functions`.

See `ES_np(<matrix>)` from **qrmtools** for another test function.

Value

`sum_of_squares()` returns an n-vector (`numeric(n)`) with the rowwise computed scaled sum of squares (theoretically integrating to 1).

`sobol_g()` returns an n-vector (`numeric(n)`) with the rowwise computed Sobol' g functions.

`exceedance()`'s return value depends on `method`:

"indicator" returns indicators whether, componentwise, `x` exceeds the threshold determined by `q`.

"individual.given.sum.exceeds" returns all rows of `x` whose sum exceeds the threshold determined by `q`.

"sum.given.sum.exceeds" returns the row sums of those rows of `x` whose sum exceeds the threshold determined by `q`.

Author(s)

Marius Hofert and Christiane Lemieux

References

Radovic, I., Sobol', I. M. and Tichy, R. F. (1996). Quasi-Monte Carlo methods for numerical integration: Comparison of different low discrepancy sequences. *Monte Carlo Methods and Applications* **2**(1), 1–14.

Faure, H., Lemieux, C. (2009). Generalized Halton Sequences in 2008: A Comparative Study. *ACM-TOMACS* **19**(4), Article 15.

Owen, A. B. (2003). The dimension distribution and quadrature test functions. *Stat. Sinica* **13**, 1-–17.

Sobol', I. M. and Asotsky, D. I. (2003). One more experiment on estimating high-dimensional integrals by quasi-Monte Carlo methods. *Math. Comput. Simul.* **62**, 255—263.

Examples

```
## Generate some (here: copula, pseudo-random) data
library(copula)
set.seed(271)
cop <- claytonCopula(iTau(claytonCopula(), tau = 0.5)) # Clayton copula
U <- rCopula(1000, copula = cop)

## Compute sum of squares test function
mean(sum_of_squares(U)) # estimate of E(3(sum_{j=1}^d U_j^2)/d)

## Compute the Sobol' g test function
if(packageVersion("copula") >= "0.999-20")
    mean(sobol_g(U)) # estimate of E(<Sobol's g function>)
```

```
## Compute an exceedance probability
X <- qnorm(U)
mean(exceedance(X, q = qnorm(0.99))) # fixed threshold q
mean(exceedance(X, p = 0.99)) # empirically estimated marginal p-quantiles as thresholds

## Compute 99% expected shortfall for the sum
mean(exceedance(X, p = 0.99, method = "sum.given.sum.exceeds"))
## Or use ES_np(X, level = 0.99) from 'qrmtools'
```

`to_array` *Compute Matrices to Arrays*

Description

Converting higher-dimensional matrices of quasi-random numbers to arrays of specific formats.

Usage

```
to_array(x, f, format = c("(n*f,d)", "(n,f,d)"))
```

Arguments

`x`	(n, fd)-matrix of quasi-random numbers to be converted.
`f`	factor $f \geq 1$ dividing `ncol{x}`.
`format`	`character` string indicating the output format to which `x` should be converted.

Details

`to_array()` is helpful for converting quasi-random numbers to time series paths.

Value

`(n * f, d)-matrix` or `(n, f, d)-array` depending on the chosen `format`.

Author(s)

Marius Hofert

See Also

`korobov()`, `ghalton()`, `sobol()`.

Examples

```
## Basic call
N <- 4 # replications
n <- 3 # time steps
d <- 2 # dimension
set.seed(271) # note: respected for the choice of 'randomize'
x <- sobol(N, d = n * d, randomize = "digital.shift") # higher-dim. Sobol'
stopifnot(dim(to_array(x, f = n)) == c(N * n, d)) # conversion and check
stopifnot(dim(to_array(x, f = n, format = "(n,f,d)")) == c(N, n, d))

## See how the conversion is done
(x <- matrix(1:(N * n * d), nrow = N, byrow = TRUE))
to_array(x, f = n) # => (n * d)-column x was blocked in n groups of size d each
```

Index