Package ‘qrng’

August 7, 2019

Version 0.0-6
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 <marius.hofert@uwaterloo.ca>
Depends R (>= 3.0.0)
Imports methods, randtoolbox, utils, copula
Suggests
Enhances
License GPL-2 | GPL-3
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-08-07 09:00:02 UTC

R topics documented:

qrng .......................................................... 2
test_functions ........................................... 4

Index 7
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 of points to be generated ≥ 2.
- d: Dimension d.
- generator: A numeric of length d or length 1 (in which case it is appropriately extended to length d). All numbers must be in \{1, ..., n\} and must be (coercible to) integers.
- randomize: A character string indicating the type of randomization for the point set
  - "none": no randomization.
  - "shift": uniform random variate modulo 1.
  - "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: A 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 scrambling methods do not (they then use 4711 as default); see sobol() from package randtoolbox.
- 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 ≤ 2^{31} − 1.
ghalton() n must be ≤ 2^{32} − 1 and d must be ≤ 360.
sobol() if randomize = "none" or randomize = "digital.shift", n must be ≤ 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


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

Description

Functions for testing low-discrepancy sequences.

Usage

sum_of_squares(u)
sobol_g(u, 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 `qrmttools` 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


Examples

```r
## 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

```r
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

```r
mean(exceedance(X, p = 0.99, method = "sum.given.sum.exceeds"))
```

## Or use ES_np(X, level = 0.99) from `qrmttools`
Index

*Topic **distribution**
  qrng, 2

*Topic **utilities**
  test_functions, 4

  cCopula, 4
  character, 2, 4
  Copula, 4
  exceedance (test_functions), 4
  ghalton (qrng), 2
  korobov (qrng), 2
  logical, 2
  matrix, 3
  numeric, 2, 5
  qrng, 2
  set.seed, 2
  sobol, 2
  sobol (qrng), 2
  sobol_g (test_functions), 4
  sum_of_squares (test_functions), 4
  test_functions, 4