About the R function: rsobol

Art B. Owen
Stanford University

January 2021

Abstract

The function `rsobol` implements a form of randomized quasi-Monte Carlo based on scrambling the digits of Sobol' sequences. This note describes that function. It also includes some theoretical background along with references for more information.

Background

In Monte Carlo (MC) studies we repeat some sampling process n times, and inspect the output to learn something. Suppose that each of those n times requires d uniformly distributed random numbers. We can then describe the process as using n points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in [0,1]^d$. Those inputs can be represented as a matrix X with n rows and d columns and each X_{ij} uniformly distributed over $[0,1]$. The i'th row of X is, of course, $\boldsymbol{x}_i$.

A lot of problems are expressed in terms of non-uniform distributions. We may then transform our uniform random variables to have other distributions, such as Gaussians with such transformations being a part of our Monte Carlo study.

In quasi-Monte Carlo (QMC) sampling we pick points $\boldsymbol{x}_i$ to more evenly sample $[0,1]^d$ than random points would ordinarily do. We measure the extent of that success via discrepancy measures on the n points. See Niederreiter (1992) or Dick and Pillichshammer (2010) for the underlying theory of how to do this and in what way the resulting n simulated outputs might be better than what we would get by MC.

A strong advantage of MC is that using statistical methods on our repeated samples, we can estimate how random sampling fluctuations among our n points have affected what we see in our n samples. For instance, we can estimate variances and construct confidence intervals to quantify uncertainty.

A drawback from QMC is that the points $\boldsymbol{x}_i$ are constructed in a deterministic way. There are no practical ways to estimate the accuracy of our answers based on those points. In randomized QMC (RQMC) we inject randomization that makes each point $\boldsymbol{x}_i$ uniformly distributed in $[0,1]^d$ while making them collectively retain the more even sampling properties of QMC points. This then supports some error estimation strategies based on repeating the whole RQMC process a few times. For a literature survey of RQMC see L'Ecuyer and Lemieux (2002). For guidance on how to use RQMC points in MC settings see Owen (2019).

One of the most widely used QMC constructions are the Sobol' points of Sobol' (1967). The function `rsobol` scrambles them using the scramble from Owen (1995). There are numerous versions of Sobol's construction. This one is based on direction numbers from Joe and Kuo (2008) using matrices from Dirk Nuyens' magic point shop as described in Kuo and Nuyens (2016). Figure 1 shows MC and QMC and RQMC points with the latter two based on Sobol' points. For properties of scrambled Sobol' points see Owen and Rudolf (2020) and references therein.

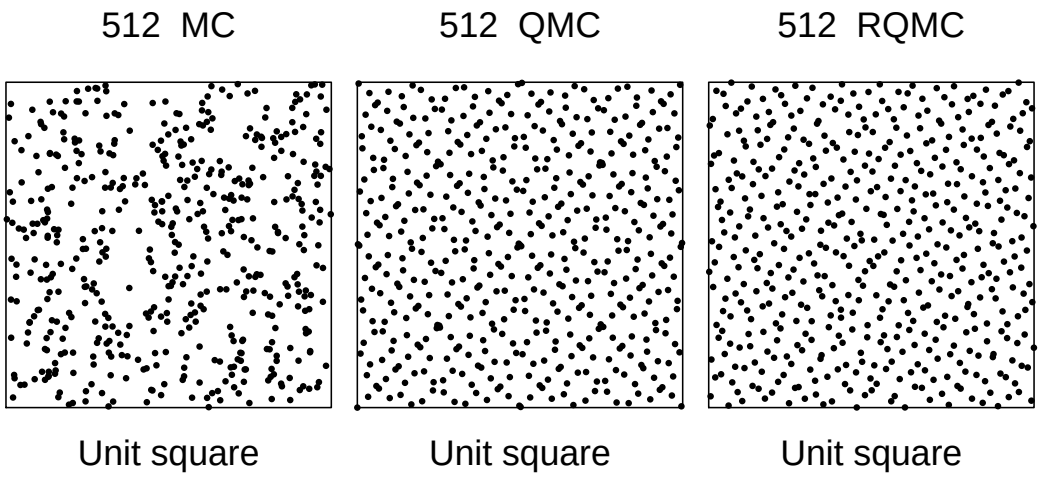

Figure 1: The left panel shows 512 Monte Carlo points in the unit square. The middle panel shows 512 Sobol' points. The right panel shows 512 scrambled Sobol' points. This image is from Owen and Rudolf (2020).

Installation

Place the files rsobol.R, fiftysobol.col, sobol_Cs.col in the folder (also called a directory) from which you plan to run R. From the command line within R run: `source("rsobol.R")`

rsobol function

Sobol' points should only be used to generate a number n of points that equals a power of 2. The `rsobol` function takes a parameter m and delivers 2^m points. Sobol' points are specially constructed for such sample sizes and a value like 2^m+1 could actually give a worse rate of convergence than using $n = 2^m$. It is not advised to use round numbers like $n = 100$ or $n = 1000$ with Sobol' points. This fact commonly surprises users. It might be because the theoretical descriptions refer to asymptotic convergence where the error decays like $O(n^{-1+\epsilon})$ for $\epsilon > 0$. The importance of special values of n is then not visible in those rates.

The definition of `rsobol` is as follows:

```
function (fn = "fiftysobol.col", m = 10, s = 5, rand = TRUE,
    type = "nestu", M = 32, seed = 20171215)
```

It will deliver $n = 2^m$ points in $[0, 1]^s$ as an $n \times s$ matrix. For example:

```
> rsobol(m=3,s=4)
          [,1]       [,2]      [,3]       [,4]
[1,] 0.2390016 0.86590940 0.8146708 0.84645296
[2,] 0.7983612 0.02943666 0.3568494 0.17595427
[3,] 0.3235185 0.37814735 0.1028197 0.42321098
[4,] 0.6154819 0.61015992 0.6009836 0.50843535
[5,] 0.1086730 0.24367359 0.7269782 0.87505254
[6,] 0.9736044 0.88991499 0.1781567 0.08907506
[7,] 0.3953411 0.71051519 0.4220558 0.25112716
[8,] 0.6965550 0.32219367 0.9632836 0.68076739
```

If you want Matousek's random linear scrambling Matoušek (1998) set `type="mato"` like this

```
> rsobol(m=3,s=4, type="mato")
          [,1]        [,2]       [,3]      [,4]
[1,] 0.1250263 0.483672959 0.49367359 0.1019782
[2,] 0.9820375 0.973604413 0.51491499 0.9281567
[3,] 0.3755636 0.645341142 0.83551519 0.6720558
[4,] 0.7153837 0.196554952 0.19719367 0.3382836
[5,] 0.1140016 0.865909396 0.06467084 0.2214530
[6,] 0.7983612 0.279436664 0.98184945 0.8009543
[7,] 0.3235185 0.003147353 0.72781969 0.5482110
[8,] 0.6154819 0.610159921 0.35098356 0.3834353
```

Matousek's random linear scrambling requires less memory than nested uniform scrambling. I believe that it has the same variance as nested uniform sampling, though to my knowledge this has not been proved in the literature. There is no central limit theorem for averages over it. There is a central limit theorem for nested uniform sampling in limited settings (where the t parameter of the Sobol' points equals zero). See Loh (2003).

If you want Sobol' points that have not been randomized, set `rand=FALSE` like this

```
> rsobol(m=3,s=4,rand=FALSE)
      [,1]  [,2]  [,3]  [,4]
[1,] 0.000 0.000 0.000 0.000
[2,] 0.500 0.500 0.500 0.500
[3,] 0.250 0.750 0.750 0.750
[4,] 0.750 0.250 0.250 0.250
[5,] 0.125 0.625 0.375 0.125
[6,] 0.625 0.125 0.875 0.625
[7,] 0.375 0.375 0.625 0.875
[8,] 0.875 0.875 0.125 0.375
```

If you want independent random replicates modify the seed parameter like this:

```
> rsobol(m=2,s=4,seed=1)
          [,1]       [,2]      [,3]       [,4]
[1,] 0.2499490 0.67250095 0.9514518 0.06973709
[2,] 0.7877009 0.41552019 0.2671969 0.59478264
[3,] 0.3897402 0.06754324 0.1969313 0.84568245
[4,] 0.5763095 0.98920148 0.5231392 0.32949544
> rsobol(m=2,s=4,seed=2)
           [,1]       [,2]       [,3]       [,4]
[1,] 0.48462954 0.48568325 0.47088599 0.85310805
[2,] 0.80974103 0.85426351 0.54045891 0.03658108
[3,] 0.04834676 0.56517998 0.95134490 0.29574076
[4,] 0.63489111 0.06583854 0.03845989 0.54901828
> rsobol(m=2,s=4,seed=3)
          [,1]       [,2]      [,3]      [,4]
[1,] 0.4542677 0.27530261 0.1082582 0.7801189
[2,] 0.6915477 0.91451606 0.7952326 0.4632242
[3,] 0.1538399 0.64318648 0.6110885 0.1442120
[4,] 0.8835448 0.02234412 0.3018011 0.7041481
```

The default call only allows $1 \le s \le 50$. If you want more dimensions set `fn="sobol_Cs.col"` as shown next. That can handle $1 \le s \le 20201$.

```
> x <- rsobol(m=3,s=51)
Error in .rsobol.sobomats(fn, m, s, M) :
  Not enough colunns in file fn = fiftysobol.col. There should be at least s = 51.
> x <- rsobol(fn="sobol_Cs.col",m=3,s=51)
> dim(x)
[1]  8 51
```

The parameter `M` is the number of bits described by the Sobol' matrices in the file called `fn`. It equals the number of columns in that matrix. It should not be changed.

The file `rsobol.R` contains numerous other functions such as helper functions for the `rsobol` function and functions that test whether the generated points work as expected. You can read them in the source code. They are given names that start with `.rsobol` so that they do not clutter up the namespace of your R session.

Disclaimer

The source code includes this notice.

```
# The software is provided "as is", without warranty of any
# kind, express or implied, including but not limited to the
# warranties of merchantability, fitness for a particular purpose
# and noninfringement. In no event shall the authors or copyright
# holders be liable for any claim, damages, or other liability,
# whether in an action of contract, tort or otherwise, arising from,
# out of, or in connection with the software or the use or other
# dealings in the software.
#
```

References

Dick, J. and Pillichshammer, F. (2010). *Digital sequences, discrepancy and quasi-Monte Carlo integration*. Cambridge University Press, Cambridge.

Joe, S. and Kuo, F. Y. (2008). Constructing Sobol' sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*, 30(5):2635–2654.

Kuo, F. Y. and Nuyens, D. (2016). Application of quasi-Monte Carlo methods to elliptic PDEs with random diffusion coefficients: a survey of analysis and implementation. *Foundations of Computational Mathematics*, 16(6):1631–1696.

L'Ecuyer, P. and Lemieux, C. (2002). A survey of randomized quasi-Monte Carlo methods. In Dror, M., L'Ecuyer, P., and Szidarovszki, F., editors, *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, pages 419–474. Kluwer Academic Publishers.

Loh, W.-L. (2003). On the asymptotic distribution of scrambled net quadrature. *Annals of Statistics*, 31(4):1282–1324.

Matoušek, J. (1998). *Geometric Discrepancy : An Illustrated Guide*. Springer-Verlag, Heidelberg.

Niederreiter, H. (1992). *Random Number Generation and Quasi-Monte Carlo Methods*. S.I.A.M., Philadelphia, PA.

Owen, A. B. (1995). Randomly permuted (t, m, s)-nets and (t, s)-sequences. In Niederreiter, H. and Shiue, P. J.-S., editors, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 299–317, New York. Springer-Verlag.

Owen, A. B. (2019). Monte carlo book: the quasi-monte carlo parts. `https://statweb.stanford.edu/~owen/mc/`.

Owen, A. B. and Rudolf, D. (2020). A strong law of large numbers for scrambled net integration. *SIAM Review*. to appear.

Sobol', I. M. (1967). The distribution of points in a cube and the accurate evaluation of integrals (in Russian). *Zh. Vychisl. Mat. i Mat. Phys.*, 7:784–802.