

Quick introduction of **randtoolbox**

Christophe Dutang and Petr Savicky

September 2009

Random simulation or Monte-Carlo methods rely on the fact we have access to random numbers. Even if nowadays having random sequence is no longer a problem, for many years producing random numbers was a big challenge. According to Ripley (1990), simulation started in 1940s with physical devices. Using physical phenomena to get random numbers is referred in the literature as true randomness.

However, in our computers, we use more frequently pseudo-random numbers. These are defined as deterministic sequences, which mimic a sequence of i.i.d. random numbers chosen from the uniform distribution on the interval $[0, 1]$. Random number generators used for this purpose receive as input an initial information, which is called a user specified seed, and allow to obtain different output sequences of numbers from $[0, 1]$ depending on the seed. If no seed is supplied by the user, we use the machine time to initiate the sequence.

Since we use pseudo-random numbers as a proxy for random numbers, an important question is, which properties the RNG should have to work as a good replacement of the truly random numbers. Essentially, we need that the applications, which we have, produce the same results, or results from the same distribution, no matter, whether we use pseudo-random numbers or truly random numbers. Hence, the required properties may be formulated in terms of computational indistinguishability of the output of the generator from the truly random numbers, if the seed is not known. The corresponding mathematical theory is developed in complexity theory, see <http://www.wisdom.weizmann.ac.il/~oded/c-indist.html>.

The best known random number generators are used for cryptographic purposes. These generators are chosen so that there is no known procedure, which could distinguish their output from truly random numbers within practically available computation time, if the seed is not known. For simulations, this requirement is usually relaxed. However, even for simulation purposes, considering the hardness of detecting the difference between the generated numbers and truly random ones is a good measure of the quality of the generator. In particular, the well-known empirical tests of random number generators such as Diehard¹ or TestU01 L'Ecuyer & Simard (2007) are based on relatively easy to compute statistics, which allow to distinguish the output of bad generators from truly random numbers. More about this may be found in section Examples of distinguishing from truly random numbers.

A simple parameter of a generator is its period. Recent generators have huge periods, which cannot be exhausted by any practical computation. Another parameter, suitable mainly for linear generators, is so called equidistribution. This parameter measures the uniformity of several most significant bits of several consecutive numbers in the sequence over the whole period. If a generator has good equidistribution, then we have a reasonable guarantee of practical independence of several consecutive numbers in the sequence. For linear generators, determining equidistribution properties may be done by efficient algebraic algorithms and does not need to really generate the whole period.

Ripley (1990) lists the following properties

- output numbers are almost uniformly distributed,
- output numbers are independent,
- the period between two identical numbers is sufficiently long,
- unless a seed is given, output numbers should be unpredictable.

¹The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness, Research Sponsored by the National Science Foundation (Grants DMS-8807976 and DMS-9206972), copyright 1995 George Marsaglia.

The statistical software R provides several random number generators described in `?RNGkind()`. The default generator is called Mersenne-Twister and achieves high quality, although it fails some tests based on XOR operation. Still, there are reasons to provide better and more recent RNGs as well as classic statistical tests to quantify their properties. The rest of this chapter is two-folded: first we present the use of RNGs through the `runif()` interface, second we present the same use with dedicated functions (not modifying base R default RNGs). See the overall man page with the command `?randtoolbox`.

1 The `runif` interface

In R, the default setting for random generation are (i) uniform numbers are produced by the Mersenne-Twister algorithm and (ii) normal numbers are computing through the numerical inversion of the standard normal distribution function. This can be checked by the following code

```
> RNGkind()

[1] "Wichmann-Hill" "Inversion"
[3] "Rejection"
```

The function `RNGkind()` can also be used to set other RNGs, such as Wichmann-Hill, Marsaglia-Multicarry, Super-Duper, Knuth-TAOCP or Knuth-TAOCP-2002 plus a user-supplied RNG. See the help page for details.

Random number generators provided by R extension packages are set using `RNGkind("user-supplied")`. The package **randtoolbox** assumes that this function is not called by the user directly. Instead, it is called from the functions `set.generator()` and `put.description()` used for setting some of a larger collection of the supported generators.

The function `set.generator()` eases the process to set a new RNG in R. Here is one short example on how to use `set.generator()` (see the man page for detailed explanations).

```
> RNGkind()

[1] "Wichmann-Hill" "Inversion"
[3] "Rejection"

> library(randtoolbox)
> set.generator("MersenneTwister", initialization="init2002", resolution=53, seed=1)
> get.description()
```

```
generator 3
$name
```

```
[1] "MersenneTwister"
```

```
$parameters
```

```
initialization      resolution
  "init2002"        "53"
```

```
$state
```

```
 [1]          624          1
 [3] 1812433254 -581806939
 [5] -1185793151   64984499
 [7] -902309212   446538473
 [9] -1665206540 -1841621738
[11] 1394803949 1021787430
[13] 2063496713 1304877364
[15] 1713639158  889001601
[17] 1651239412 1450863289
[19]   745575081  361057727
[21] -2006195346 1463387568
[23] -2045478934  26637982
[25]  204036717 1655702041
[27] 1329048465 2092351466
[29] 1681619666 -1074306981
[31] 1301783610  626286181
[33]  294669048 -757838856
[35] -1035449048 -1744866023
[37] 1160881866  308703547
[39]  295714668  35508674
[41] 1599247281  376272024
[43] -1128507359 1852735737
[45] -614098429  612352556
[47] -1534777463 -478216955
[49]  699140493 1087846865
[51]  394927937 2063539671
[53]  645417889 -1957298247
[55] -521799684  678121169
[57] -1287982676 1163491294
[59] -1735679436  543155592
[61] -1100785949 -1831423999
[63] -419820436  475483913
[65] -587399220 -413158421
[67] 1264657097  208126250
[69] 1802809301  367907560
[71] -1861591603 -1443640847
[73] -1914259418 -1383208324
[75] -51580417 -2065738570
[77]  828161871 -1423851145
[79]  990638198  178193628
[81] 1012573979 1223581943
```

[83]	-961943713	1901888414
[85]	-381090546	-1126304907
[87]	656194888	1553610174
[89]	466840498	686407570
[91]	280737523	-1818478279
[93]	1272981410	-1105535317
[95]	-1000257014	1564477163
[97]	-161745743	823708826
[99]	880616227	1730254897
[101]	335723347	2123911971
[103]	344194767	119099153
[105]	-1379710180	-955141826
[107]	-1770024326	1191117250
[109]	-891155110	-305994359
[111]	-1719572001	-222230113
[113]	663832315	808080503
[115]	724042340	-1328777754
[117]	-1795324057	-985761715
[119]	1915303227	72616536
[121]	387525935	-1503266045
[123]	-2104061730	-554638522
[125]	831297460	-544002432
[127]	-2104855252	899144100
[129]	-1948409293	-443271467
[131]	-1398003473	1548614403
[133]	-618259891	2050891594
[135]	-129074148	1883017153
[137]	-1626179769	50330561
[139]	2063572142	1853585557
[141]	1716111087	-1357718926
[143]	1650859709	-1612661574
[145]	565243175	-372740109
[147]	-812934591	-1485885796
[149]	2099376873	230358556
[151]	1065827745	196966939
[153]	-1026121666	-669459031
[155]	1477799595	-145513556
[157]	-1537131610	-1262269360
[159]	-2094858505	-873286585
[161]	-149585037	-689714224
[163]	1186485728	-774485145
[165]	-1214233833	-407653139
[167]	-264519541	1699987022
[169]	1393253586	1710066407
[171]	710337383	-540354739
[173]	-1553878927	337455371
[175]	1304761604	-702285657
[177]	-1195582109	-291290891

[179]	317081535	997754381
[181]	480565460	-488701864
[183]	1068029852	776179010
[185]	470617537	-641091875
[187]	-2021395377	1055365147
[189]	1317172834	-880234293
[191]	-1459566683	28845217
[193]	631741764	-1960415084
[195]	-729501201	1225096926
[197]	1277781438	-1878959073
[199]	1268768054	-1544178055
[201]	267768398	-2119583858
[203]	268654341	-1744436541
[205]	-1323343888	1666669894
[207]	1934871760	509782083
[209]	-1496498626	-1460950404
[211]	-1800818041	1965005899
[213]	-1641921531	-1977772393
[215]	1297426078	916214929
[217]	-1327106292	-2058160290
[219]	-1818242011	128488253
[221]	-17253140	-1278774745
[223]	1690883702	1329810641
[225]	593010415	-1953653717
[227]	1754238478	1242698701
[229]	-2142372769	2103269013
[231]	926178633	647225267
[233]	-51180154	1489208161
[235]	-1106168375	1327553793
[237]	-650366485	684513652
[239]	-1688412239	-1589637747
[241]	-1737498278	1294205096
[243]	70104222	-1274883768
[245]	2015571237	-1526393816
[247]	401698695	-1482604487
[249]	328919870	984940142
[251]	1653817439	471643152
[253]	538942283	2040555667
[255]	1211982999	1663497772
[257]	-1353173568	-1293940598
[259]	313271977	-650464593
[261]	-1871017249	-1665921227
[263]	-844140360	44600781
[265]	-1661098008	-27952550
[267]	-90052826	1955987363
[269]	-1704358411	2120168063
[271]	1460034243	258056600
[273]	-601417209	779446436

[275]	902696389	-66265909
[277]	-1129176069	-816352431
[279]	1500865135	905884796
[281]	-612920829	-1857119464
[283]	-1699079077	-150482633
[285]	1299603103	648536946
[287]	1762836247	-29218100
[289]	950840266	-1365974574
[291]	2051369009	2071186450
[293]	1164619682	210405235
[295]	1296628868	-1869492577
[297]	-211580392	1978331343
[299]	-1104068497	602128683
[301]	2003319330	1043377147
[303]	756690484	24776626
[305]	1835824233	1156421176
[307]	2125448878	1333136189
[309]	607751135	-39352529
[311]	-56434287	-1711791664
[313]	230472465	-1257707539
[315]	1546348932	-1757687885
[317]	110471952	520621708
[319]	63613561	-1451293701
[321]	775036	1899744556
[323]	1168115970	-1609880975
[325]	-884716638	-1143865143
[327]	634647644	-655841902
[329]	-950342532	1525171811
[331]	1878800371	-938437180
[333]	-618424370	602053165
[335]	-1608259058	-591412214
[337]	-540005924	-324936373
[339]	1749014201	-903860246
[341]	-1816815296	2121779806
[343]	-1658277936	769835312
[345]	-64427705	1909812524
[347]	417081626	-1198447972
[349]	387659697	-530468047
[351]	-842041833	-476689598
[353]	-1286046972	15253694
[355]	1479260759	-1873638576
[357]	-2074223939	38831551
[359]	1032912064	-894011098
[361]	-1932158464	-306260430
[363]	1950464958	-1046394171
[365]	1225815945	1211036180
[367]	346407094	-427790532
[369]	1257086026	-1569731065

```
[371] -1451231638 -147726214
[373] 1729974832 1256499145
[375] -528991395 784776076
[377] -6689869 -391434776
[379] -863444432 -1502377319
[381] -1358978142 -758370404
[383] -781983176 605476293
[385] 1774961976 981422589
[387] 822525778 -951427364
[389] 422954622 1323482938
[391] -1771501876 -1548357940
[393] 1664448205 272567300
[395] 711582493 -669245189
[397] -679101597 950619756
[399] -1430798807 108006277
[401] -318653944 680217319
[403] 173747636 291134870
[405] 198587329 595310009
[407] 941470866 -1856478928
[409] 1681923153 1654783272
[411] -763178042 -145425581
[413] -1372260309 684907209
[415] -1178278934 -1006824410
[417] -341589704 -962539289
[419] 1400401813 -549045498
[421] 1701705628 -550455403
[423] 1838265811 -980934784
[425] -400127146 -484935887
[427] 181324387 983160249
[429] 1444959400 -458303143
[431] -1262293969 310789231
[433] -593401734 1407580781
[435] -1783391667 -1181144611
[437] 1777261998 -2086068545
[439] 106383174 -1333946796
[441] 995776421 -988880175
[443] -2113937261 -1994902545
[445] 1909543740 -271811123
[447] 1671619075 -2143011192
[449] 237668401 -1090456043
[451] 1303668692 -426707509
[453] -1557069397 -203941263
[455] -1417186625 134376279
[457] 398912026 863520778
[459] -582498373 -851753630
[461] -2111157744 -1697587994
[463] 349776833 274697715
[465] -28373586 -12780527
```



```
[467] -764209429 520237914
[469] -925929899 -2009296958
[471] 387086485 618942879
[473] 219892882 2008897906
[475] -2001217736 -1387530820
[477] -441670703 327550390
[479] 1558751403 2125694704
[481] 1822570484 -1884999031
[483] 436622776 -1603843206
[485] 1080819771 -1336859962
[487] -1627808455 2117901613
[489] 440045635 -433862825
[491] -720004595 -1084668048
[493] 1368601573 -1860927776
[495] 86704919 -666859263
[497] 1909858745 227461000
[499] -1764457831 838433817
[501] 730224848 1060658180
[503] 1318482825 233266846
[505] -1942166451 2086493219
[507] -468611741 -1120589606
[509] 1455208243 1356597942
[511] 663563056 -1793147922
[513] -81432037 1585241464
[515] 873997246 -1697068552
[517] 427064229 1587746589
[519] 259660817 1688808891
[521] -129132951 1359025114
[523] 2013923952 -1331455585
[525] -1391746564 356112706
[527] 501549847 1609412897
[529] 1685128111 -1655663690
[531] 700554261 914150235
[533] 2010650618 2029243163
[535] -1248457385 715702687
[537] -2088010542 -1249669080
[539] -1372300117 -1797389881
[541] -1293147692 706666890
[543] -2019043441 -1200782913
[545] -1513269584 -1002014630
[547] -56353218 278500659
[549] 1440033346 1552714131
[551] 336554687 -1452386687
[553] -2039922986 -2114895924
[555] 99970159 2078552309
[557] 1172694639 1359399314
[559] 546452524 349053834
[561] -1222712927 -1251720577
```

```
[563] -980540798 1594992663
[565] -712697631 2114045278
[567] 585873328 840739494
[569] -819188811 1506518790
[571] -286480644 229989333
[573] -712689084 363921215
[575] -702124776 1833533669
[577] 708173875 564248927
[579] 853943228 -2012235922
[581] -1420809249 -316304011
[583] -1962270765 1354524859
[585] 58121641 1445193461
[587] 1936635021 -920639098
[589] -829714236 385589199
[591] 1819596280 912895627
[593] 1877426726 733280947
[595] 2004202992 -983186585
[597] -562914105 309903272
[599] 97290141 -1349547961
[601] -378490224 1326195031
[603] -554029241 -690222034
[605] -661658340 -902037865
[607] 1257547457 251825182
[609] -976267211 847033774
[611] 137350663 1716455973
[613] 546850455 -67392777
[615] -1250752343 -2035093283
[617] -1852219038 -1337995960
[619] -2096194917 1269686727
[621] -1646851191 1339159363
[623] 1473334647 -1908295684
[625] 2069268389
```

```
$authors
```

```
[1] "M. Matsumoto, T. Nishimura, 1998"
```

```
> RNGkind()
```

```
[1] "user-supplied" "Inversion"
[3] "Rejection"
```

```
> runif(10)
```

```
[1] 0.00011 0.30233 0.09234 0.02739
[5] 0.87814 0.95789 0.68650 0.83463
[9] 0.98886 0.13003
```

Random number generators in **randtoolbox** are represented at the R level by a list containing mandatory components `name`, `parameters`, `state` and possibly an optional component `authors`. The function `set.generator()` internally creates this list from the user supplied information and then runs `put.description()` on this list in order to really initialize the generator for the functions `runif()` and `set.seed()`. If `set.generator()` is called with the parameter `only.dsc=TRUE`, then the generator is not initialized and only its description is created. If the generator is initialized, then the function `get.description()` may be used to get the actual state of the generator, which may be stored and used later in `put.description()` to continue the sequence of the random numbers from the point, where `get.description()` was called. This may be used, for example, to alternate between the streams of random numbers generated by different generators.

From the `runif()` interface, you can use any other linear congruential generator with modulus at most 2^{64} and multiplier, which is either a power of 2 or the product of the modulus and the multiplier is at most 2^{64} . The current version of the package also allows to use Well-Equidistributed Long-period Linear generators (WELL).

To get back to the original setting of RNGs in R, we just need to call `set.generator` with default RNG.

```
> set.generator("default")
> RNGkind()
```

```
[1] "Mersenne-Twister"
[2] "Inversion"
[3] "Rejection"
```

2 Dedicated functions

The other way to use RNGs is to directly use dedicated functions. For instance to get the previous example, we can simply use

```
> setSeed(1)
> congruRand(10, mod = 2^31-1, mult = 16807, incr = 0)
```

```
[1] 7.8e-06 1.3e-01 7.6e-01 4.6e-01
[5] 5.3e-01 2.2e-01 4.7e-02 6.8e-01
[9] 6.8e-01 9.3e-01
```

where `setSeed` function initiates the seed for RNGs implemented in **randtoolbox** and `congruRand` calls the congruential generator.

They are many other RNGs provided by RNGs in addition to linear congruential generator, WELL generators, SFMersenne-Twister generators and Knuth-TAOCP double version. See `?pseudo.randtoolbox` for details.

This package also implements usual quasi random generators such as Sobol or Halton sequences (see `?quasi.randtoolbox`). See the second chapter for an explanation on quasi RNGs.

References

- L'Ecuyer, P. & Simard, R. (2007), 'Testu01: A c library for empirical testing of random number generators', *ACM Trans. on Mathematical Software* **33**(4), 22. 2
- Ripley, B. D. (1990), *Stochastic Simulation*, John Wiley & Sons. 2