《应用概率统计》版权所有

应用简报

S-PLUS 与 Visual C++ 随机数发生器性质的比较

刘 年 青

(北京大学光华管理学院，北京， 100871)

摘 要

本文通过对统计软件 S-PLUS 和 Visual C++6.0 自带的随机数发生器产生的一定数量的随机数进行参数检验、均匀性检验、独立性检验以及周期检验后，发现 S-PLUS 产生的随机数列均匀性、独立性以及周期明显地优于 VC++.

关 键 词： 均匀随机数发生器，参数检验， χ^2 检验，序列检验，列联表检验，周期检验.

学 科 分 类 号： O213.2.

§1. 引 言

一个好的随机数发生器不但要有很好的均匀性和独立性，同时还要有足够长的周期 [4]. 但在目前对随机数发生器进行检验的文献中，绝大部分只就均匀性与独立性进行检验，而对周期的检验很少涉及. 然而事实上，很多实际问题对周期是有比较高的要求的 [2]. 笔者在对 S-PLUS 的均匀随机数发生器 runif( ) 和 VC++ 自带的均匀随机数发生器 rand( ) 产生的随机数进行均匀性、独立性和周期检验时便发现 S-PLUS 的随机数发生器产生的随机数的均匀性、独立性和周期明显地优于 VC++.

§2. 均匀随机数的检验

均匀随机数的检验包括参数检验，均匀性检验，列联表检验，周期检验. 其中均匀性检验常用的两种方法是 χ^2 检验和序列检验. 注意到在进行以上除周期检验外的其他检验时都要求被检验的随机数列 $\{r_i\}$ 是由 $[0,1]$ 上的随机数发生器 (即该发生器产生的随机数在 $[0,1]$ 区间上) 产生的. 文献 [1] 对参数检验、均匀性检验 (包括 χ^2 检验和序列检验) 、列联表检验都进行了详尽的说明，因而在此仅对大部分文献没介绍过的周期检验进行较详细的介绍.

一般来说，算法确定以后，从理论上可以给出随机数发生器的周期. 但我们必须注意到随机数发生器的周期是依赖于初值的，并且有些发生器的算法是比较复杂的，因而从理论的角度给出随机数发生器的周期将会是一件很繁琐的事情，在有些情况下甚至不可能. 而从实证的角度来检验某些初值下随机数发生器的周期在任何情况下都可行并且是比较方便的，因而本文是从实证的角度对 runif( ) 和 rand( ) 在某些初值下的周期进行检验的. 注意到 rand( ) 所使用的

本文 2003 年 3 月 13 日收到， 2006 年 6 月 26 日收到修改稿.

算法 — 乘同余法使用的是单初值的递推公式 (即只需一个初值的递推公式), 而且 runif( ) 这个发生器也是单初值的，因而这里只讨论单初值递推式随机数发生器周期的检验问题.

当随机数发生器是使用单初值的递推公式来产生随机数列时，它的周期是指由该发生器所产生的随机数列重复数之间的最短长度 (即循环长度). 因而该种发生器周期的检验想法很简单：当初值为同一数值时，产生 T 个随机数两两互异，但产生 $(T+1)$ 个随机数便有重复数值出现，则该随机数发生器在该初值下的周期为 T. 因而，当由单初值递推式随机数发生器在某一初值下产生的 S 个随机数至少有两个随机数数值相同时，我们便认为该发生器在该初值下的周期小于 S.

§3. S-PLUS 随机数发生器和 Visual C++ 自带随机数发生器性质检验结果的比较

S-PLUS 的随机数发生器 runif$(n, \min, \max)$ 产生的是 $[\min, \max]$ 区间上的 n 个均匀伪随机数；Visual C++ 的随机数发生器 rand( ) 产生的随机数列是一组在 [0,RAND_MAX] 上的均匀简单整数样本[5], 因而可认为 rand( )/RAND_MAX 产生的随机数列近似为一组 $[0,1]$ 上的均匀简单样本. 我们在进行参数检验、均匀性检验和列联表检验时, 用 runif$(n,0,1)$ 和 rand( )/RAND_MAX 分别产生样本 $\{r_i\}$ $(i=1,2,\cdots,n)$, 且两个随机数发生器的种子 seed 都分别取 10、100、5000, n 分别取 100、1000、10000, $\alpha=0.05$.

表 1 给出的是 S-PLUS 与 VC++ 两种软件随机数发生器参数检验的主要结果，其中, u_i $(i=1,2,3)$ 为文献 [1] 中定义的检验统计量. 表中斜体部分便是检验中被拒绝的情形，从表中可以看出： S-PLUS 的均匀随机数发生器 runif( ) 在进行参数检验的拒绝率仅为 7.4%, 而 VC++ 的均匀随机数发生器 rand( ) 的拒绝率却高达 22.2%.

表 1 参数检验的主要结果

seed	n	u_1		u_2		u_3	
		S-PLUS	VC++	S-PLUS	VC++	S-PLUS	VC++
10	100	-0.051792	-1.514491	-0.190982	-1.783787	-0.563641	-1.525992
	1000	1.461216	1.741697	1.407949	1.874068	-0.102958	0.643461
	10000	*3.028139*	*2.184045*	*2.690507*	*2.414664*	-1.068423	1.146554
100	100	-0.030141	*-2.045085*	-0.288017	*-2.245486*	-1.035436	-1.528967
	1000	0.849307	*-2.332241*	0.674460	*-2.508107*	-0.617016	-1.192008
	10000	1.795676	-0.664429	1.450275	-1.096664	-1.189576	-1.818269
5000	100	-0.917817	1.285112	-1.225893	1.216376	-1.443065	-0.296358
	1000	1.789849	1.914771	1.727398	1.826248	-0.135730	-0.240507
	10000	-0.843807	-0.585747	-0.639943	-0.875086	0.700316	-1.235590

表 2 和表 3 给出的是 runif( ) 和 rand( ) 均匀性检验的主要结果，表中斜体部分便是检验中被拒绝的情形. 表 2 给出的是 χ^2 检验的结果，其中， m、V 均为文献 [1] 中定义的量. 表 3 给

《应用概率统计》版权所有

出的是序列检验的结果， k、V 均为文献 [1] 中定义的量.

从表 2 可知， runif( ) 在进行 χ^2 检验的拒绝率为 0, 而 rand( ) 的拒绝率仍高达 22.2%; 由表 3 可看出： runif( ) 在进行序列检验的拒绝率为 0, rand( ) 的拒绝率高达 22.2%. 因而， runif( ) 产生的伪随机数列的的均匀性远远地好于 rand( ) 产生的伪随机数列的均匀性.

表 2　χ^2 检验的主要结果

seed	n	m	V	
			S-PLUS	VC++
10	100	10	6.4	7.6
	1000	10	6.28	12.72
	10000	20	26.392	*30.96*
100	100	10	3.6	16.4
	1000	10	6.26	11.32
	10000	20	23.248	*33.976*
5000	100	10	11.8	5.8
	1000	10	8.96	6.12
	10000	20	19.088	14.876

表 3　序列检验的结果

seed	n	k	V	
			S-PLUS	VC++
10	100	4	20	15.2
	1000	4	10.24	18.016
	10000	5	25.86	27.97
100	100	4	6.24	22.88
	1000	4	10.688	18.336
	10000	5	23.8	*44.245*
5000	100	4	18.08	10.4
	1000	4	10.912	11.456
	10000	5	18	*38.52*

表 4 给出的是两随机数发生器列联表检验的主要结果, 表中的斜体部分便是检验中被拒绝的情形. 在这里， e 分别取 1 和 50, 当 $n = 100, 1000$ 时， $m = 4$; 当 $n = 10000$ 时， $m = 5$. e、m、V 在文献 [1] 中给出了它们的定义.

表 4 告诉我们： runif( ) 在进行列联表检验时的拒绝率为 11.1%, 而 rand( ) 的拒绝率为 22.2%. 故 runif( ) 产生的随机数列的独立性明显地好于 rand( ).

表 5 给出的是两随机数发生器周期检验的主要结果. 其中， seed 与 n 的含义同前， l 为随机数样本中各个重复数字重复次数的总和. 首先来看 rand( ) 检验的结果. 从表 5 可以看出在 seed = 70、600、5500、10000, 当 $n = 200$ 时，由 rand( ) 产生的随机数列是两两互异的; 而当

《应用概率统计》版权所有

$n = 250$ 时，随机数列中便有重复数字出现；当 $n = 10000$ 时，随机数样本中各个重复数字重复次数的总和大于 1300, 在样本中所占比例大于 13%; 当 $n = 15000$ 时，样本中各重复数字重复次数的总和约为 3000, 约占样本的 20%. 这便说明在 seed = 70, 600, 5500, 10000 时，随机数发生器 rand( ) 的实际周期的大小介于 200 至 250 之间，且若产生的随机数的个数为 10000–15000 时，由于重复率太高， rand( ) 已不再适于作随机数发生器了，即 rand( ) 不适合运用于大中型问题的解决当中．而对于 runif( ) 来说 seed = 70, 600, 5500, 10000 时，其周期均不小于 15000, 达到了中长周期．事实上，对于 seed 取其他数值的情形，也有类似的结果．

表 4　列联表检验的主要结果

seed	n	m	$e = 1$ 时 V 的值		$e = 50$ 时 V 的值	
			S-PLUS	VC++	S-PLUS	VC++
10	100	4	12.688642	6.015385	*16.816185*	3.261426
	1000	4	13.599536	8.554775	10.782687	5.233897
	10000	5	16.246037	12.835251	21.870622	15.747633
100	100	4	9.849518	7.649976	15.132950	*16.963508*
	1000	4	5.915284	7.802220	9.366922	0.607789
	10000	5	20.928791	*29.610597*	10.662926	*27.790181*
5000	100	4	5.317423	13.690150	5.262222	15.120126
	1000	4	13.790083	6.467714	*22.475862*	*22.00905*
	10000	5	8.055737	24.351352	14.593740	16.226306

表 5　周期检验的结果

seed	n	l		seed	n	l	
		S-PLUS	VC++			S-PLUS	VC++
70	200	0	0	5500	200	0	0
	250	0	1		250	0	1
	10000	0	1368		10000	0	1356
	15000	0	2968		15000	0	2960
600	200	0	0	10000	200	0	0
	250	0	1		250	0	3
	10000	0	1369		10000	0	1388
	15000	0	2922		15000	0	2959

§4. 结　　论

从 3 中所列的各项检验的主要结果易看出：S-PLUS 的随机数发生器 runif( ) 产生的随机数列的均匀性和独立性明显地优于 VC++ 的随机数发生器 rand( ), runif( ) 的周期远大于 rand( ) 的周期．由于 Visual C++ 经常作为大中型问题的开发工具，且它所附的随机数发生器 rand( )

《应用概率统计》版权所有

也频繁地直接用于解决实际问题的程序中，因而将 VC++ 的随机数发生器更新为独立性与均匀性更好，周期足够大的随机数发生器对于运用 VC++ 解决许多大中型问题具有极为重要的现实意义．事实上，比 VC++ 现有的随机数发生器 rand( ) 独立性与均匀性更好，周期足够大的随机数发生器早在 20 世纪五、六十年代就已经提出来了，且其中许多发生器用程序实现起来也是很简单，只需短短的几行，例如积式发生器便是其中的一个．

致谢 本文是作者在南开大学攻读本科时由王公恕教授指导而完成的，本人在此向他致以诚挚的谢意．同时还要感谢加拿大滑铁卢大学统计与精算系的陈家骅教授在访问南开大学时对本文所提出的宝贵意见．

《应用概率统计》版权所有

参 考 文 献

[1] 高惠璇， 统计计算, 北京大学出版社，北京， 1995.
[2] 黄宏新，用变分 Monte Carlo 方法直接计算相关能， 湖南师范大学自然科学学报, **17(2)**(1994), 59–63.
[3] 盛骤，谢式千，潘承毅， 概率论与数理统计, 高等教育出版社，北京， 1989.
[4] 王尧，一种新的随机数生成法， 交通与计算机, **17(3)**(1999), 67–69.
[5] [美]Harvey M.Deitel, Paul James Deitel 原著，邱仲潘等 译， C++ 大学教程 (第二版), 电子工业出版社，北京， 2002.
[6] Johnson, N.L. and Kotz, S., *Continuous Univariate Distributions, Vol. 2*, Houghton-Mifflin, Boston, 1970.