Statistical Computing Project 1

Mario Ibanez

January 23, 2016

Introduction

The purpose of this report is to evaluate how well the pseudorandom number generator

$$X_{n+1} = 7^5 X_n \mod(2^{31} - 1)$$

generates numbers that appear to come from a standard uniform distribution. The Kolmogorov-Smirnov test will be used in this report to evaluate this pseudorandom number generator.

Discussion

An important definition is that of the sample distribution function, also known as the empirical cumulative distribution function (ECDF). With the sample of random numbers sorted in increasing order, and r denoting the rank in the sequence and x_r being the r^{th} smallest value in the sequence, the defintion is:

$$S_n(x) \begin{cases} 0 & \text{for } x < x_1 \\ \frac{r}{n} & \text{for } x_r \leq x < x_{r+1} \\ 1 & \text{for } x_n \leq x \end{cases}$$

The test compares the empirical cumulative distribution function of the randomly generated numbers to a given CDF, in this case, the standard uniform CDF. Specifically, the test is:

$$H_o : X \sim U(0,1)$$

$$H_a : \text{not } X \sim U(0,1)$$

The null hypothesis is rejected if the test statistic, $sup_x|S_n(x) - F(x)|$ is less than a specific value. Since for the standard uniform distribution, $F(x) = x$, we can simplify the test statistic. More explicitly:

$$\text{If } sup_x|S_n(x) - x| < \frac{1.6276}{n}, \text{ fail to reject } H_o, \text{ otherwise reject } H_o$$

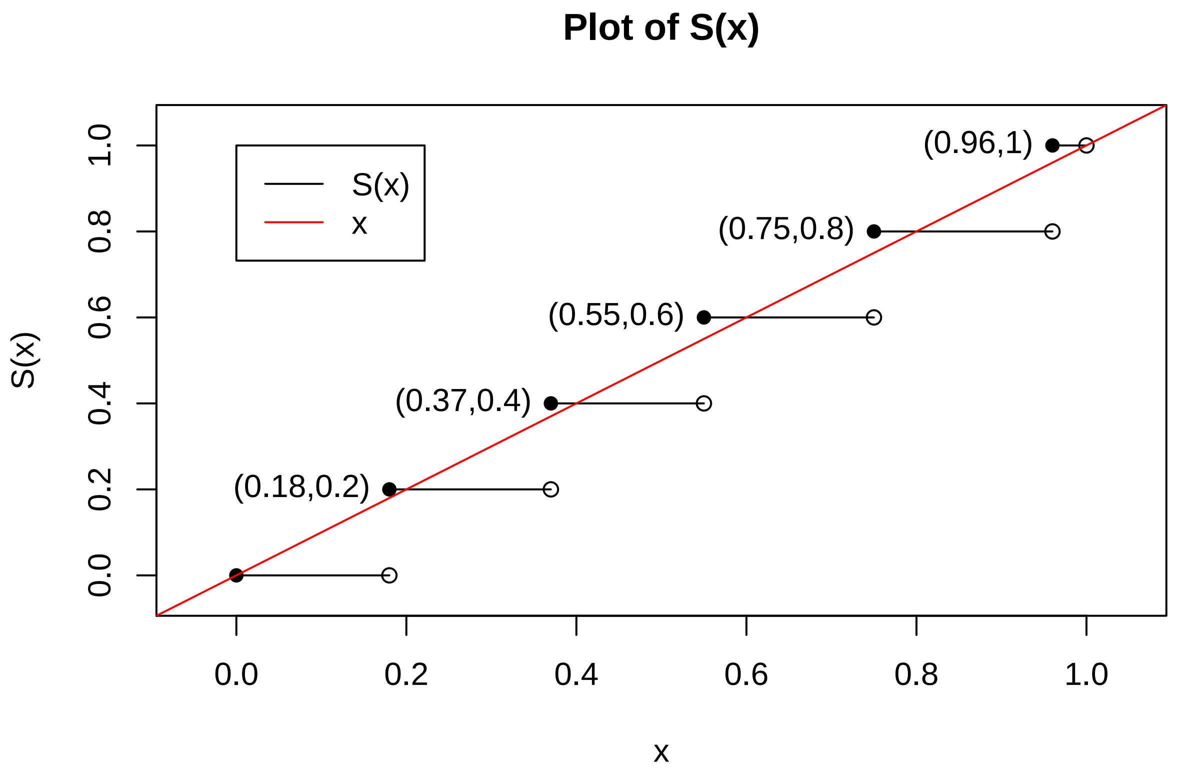
In the rule above, n is the length of the sequence of random numbers. Using this decision rule, the type 1 error rate is $\alpha = 0.01$.

Illustrative Example

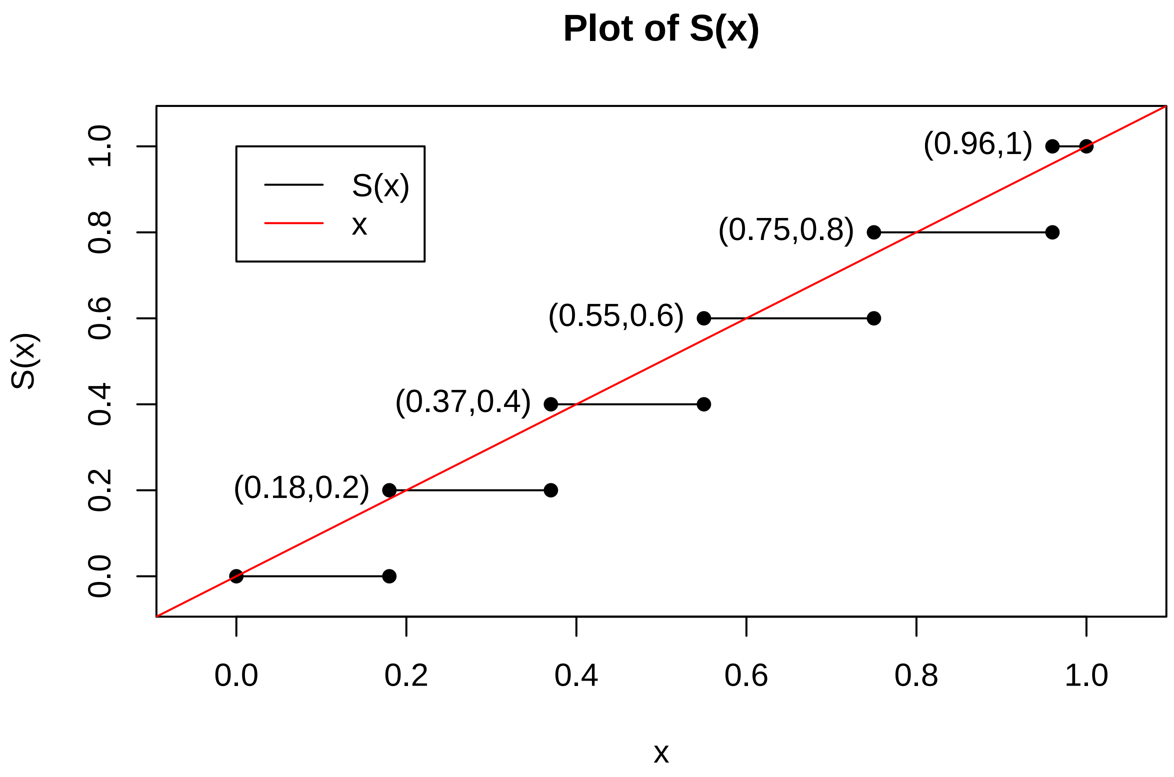
An example is being included in this report in order to show that care needs to be taken when calculating the test statistic. Suppose we have a five randomly generated numbers from the standard uniform distribution, shown in the table below along with their function values according to the defintion above.

x	$S(x)$	$\lvert S(x) - x\rvert$
0.18	0.2	0.02
0.37	0.4	0.03
0.55	0.6	0.05
0.75	0.8	0.05
0.96	1.0	0.04

It may be tempting to conclude that $sup_x|S_n(x) - x| = 0.05$ based on the third column of the table. However, by looking at a plot of the actual step function $S_5(x)$, it is easy to see that the supremum is actually greater than 0.05.



Visually it is easy to see that the value of $sup_x|S_n(x) - x|$ is greater than the differences shown in the table. This is because the supremum must be taken over all values of x, not just the five values in the sequence of random numbers. Visually, and computationally, it helps to ignore the fact that the function is only right continuous, and put in additional values. Then the task of finding the supremum amounts to finding the greatest distance between a line segment endpoint and the red line, the identity function.



With the endpoints filled in, we now look at all the vertical distances between line segment endpoints and the red line. Below is a table with the needed information:

x	$S(x)_1$	$S(x)_2$	$\|S(x)_1 - x\|$	$\|S(x)_2 - x\|$
0.18	0.0	0.2	0.18	0.02
0.37	0.2	0.4	0.17	0.03
0.55	0.4	0.6	0.15	0.05
0.75	0.6	0.8	0.15	0.05
0.96	0.8	1.0	0.16	0.04

We can now conclude correctly that the test statistic would be equal to 0.18. This example serves to illustrate the procedure that will be implemented in code in order to calculate the test statistic.

Generating and testing a single sequence

In order to test whether a single sequence generated by our random number generator, we can choose an arbitrary seed value for X_0. Let $X_0 = 6$. The R code below along with the comments show the steps for the procedure as well as perform the test.

```r
# 1) Initialize vector and set seed
random_nums <- c(rep(0,1000)); random_nums[1] <- 6

# 2) Loop to generate remaining numbers
for(i in 2:1000){
  random_nums[i] <- ( 7^5 * random_nums[i-1] ) %% (2^31 - 1)
}

# 3) Sort and divide by 2^31 - 1
random_nums <- sort(random_nums/(2^31 - 1))

# 4) S(x) values to use for Kolmogorov test
comparison_Snx <- data.frame(low=seq.int(0, 999)/1000,
                             high=seq.int(1,1000)/1000)

# 5) Find supremum
max_difference <- max(max(abs(random_nums - comparison_Snx$low)),
                      max(abs(random_nums - comparison_Snx$high)))

# 6) Print result of the test
if(max_difference < 1.6276/sqrt(1000)){
  print("PASS")
} else {
  print("FAIL")
}
```

```
## [1] "PASS"
```

The result of this test was "Pass". In other words, using the seed value $X_0 = 6$, this specific sequence of 1000 pseudorandom numbers passed the Kolmogorov test. Aside from the fact that it is known how these numbers were generated, it would be reasonable to assume that they were generated from a standard uniform distribution.

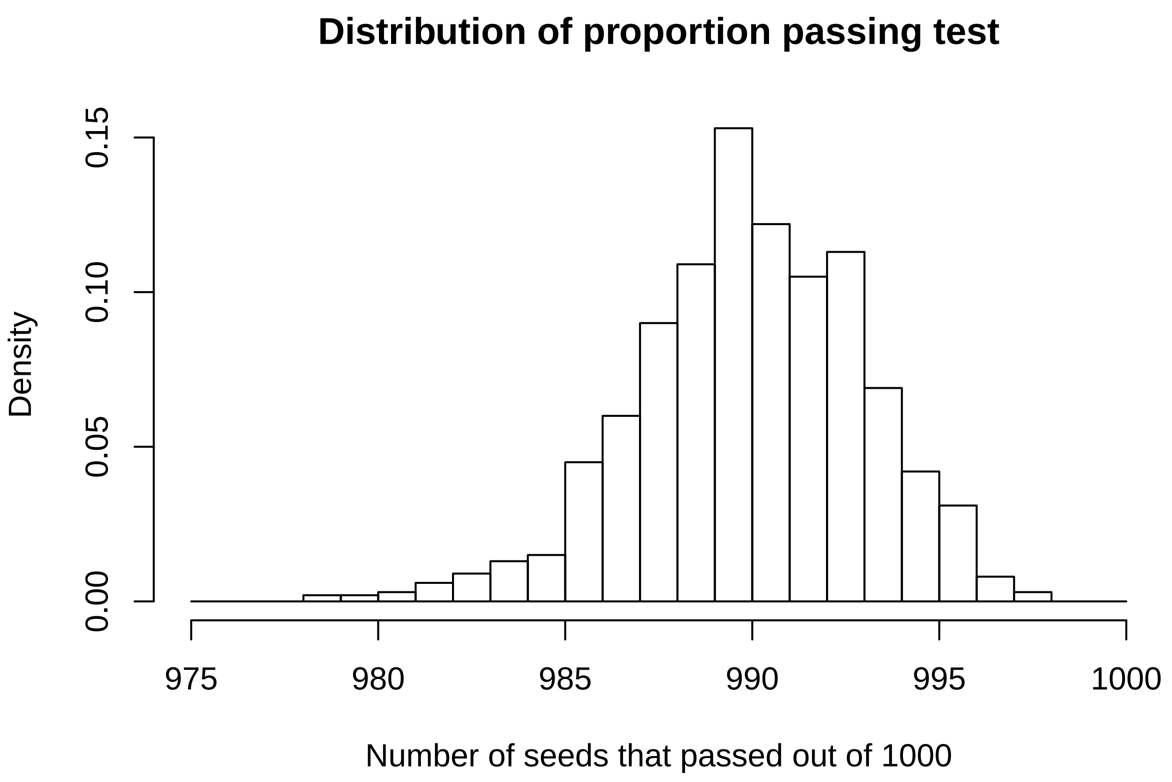
What about the other 2,147,483,646 sequences?

Though the computer is very quickly able to use a seed to generate 1000 numbers and then perform the Kolmogorov test, checking every possible seed is a different matter. There is material in abstract algebra and the theory of finite fields that applies to this very topic in this project. We know that it would not be ideal if our sequences of 1000 numbers contained a high number of repeated values. With the correct choice of multiplier and modulus in the random number generator, repetition can be eliminated (as long as the sequence length is less than the size of the modulus).

Testing the first 1,000,000 seeds

Though it is difficult to test every seed, it is not difficult to test a million different seeds. For simplicity, the first million seeds can be checked (from 1 to 1 million). This set of seeds is representative of the population of seeds. The R code in the appendix shows how this large scale test is carried out.

Testing the first million seeds was done a thousand seeds at a time. For each group of thousand seeds, the number that passed the test was recorded. Below is a histogram of the number of seeds out of a thousand that passed the test, for each group of thousand from one to one million.



What the histogram shows is that the majority of the time, between 985 and 995 out of 1000 seeds will pass the Kolmogorov test. In a sense, the way that the seeds were grouped is arbitrary or irrelevant (it is not meaningful to compare the first thousand seeds to the second thousand seeds). The histogram does give an indication though that there are few “pockets” of seeds that fail the test. In almost each of the 1000 sets of 1000 seeds, at least 980 seeds will pass the test. Whether a seed is chosen in the range (1, 1000) or in (402,001, 403,000), there is a high chance that a good sequence of pseudoandom numbers will be generated.

Overall, in the first million seeds, 99.0441% of seeds passed the test. A 95% confidence interval for the population proportion of seeds that pass the test is [0.9902479 0.9906303] using the formula

$$\hat{p} \pm z_{\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

This confidence interval is based on the assumption that the first million seeds serve as a representative sample of the population.

Conclusion

Given these results, it is reasonable to assume that this is a good psuedorandom number generator. Of the first million seeds, 990,441 create sequences that pass the Kolmogorov test. It would be interesting to test all $2^31 - 1$ seeds, though it would require more computing power, time, or a different program or algorithm.

Appendix

Below is the R code used to test the first million seeds.

```r
# knitr library
library(knitr)
comparison_value <- 1.6276/sqrt(1000)

# Generates 1000 values from standard normal with given seed
generator <- function(seed, multiplier=16807, mod=2147483647, N=1000, sorted=TRUE){
  # Initialize vector
  vector <- c(rep(0, times=N))
  vector[1] <- seed %% mod
  # Loop to create rest of the vector
  for(i in 2:N){
    vector[i] <- (multiplier * vector[i-1]) %% mod
  }
  # Divide values so they're between (0,1), return
  vector <- sort(vector/mod)
  return(vector)
}

# Sn(x) values to compare against F(x) = x
comparison_Snx <- data.frame(low=seq.int(0, 999)/1000,
                             high=seq.int(1,1000)/1000)

# Kolmogorov function
kolmogorov <- function(random_vec, comp_value=comparison_value){
  max_difference <- max(max(abs(random_vec - comparison_Snx$low)),
                        max(abs(random_vec - comparison_Snx$high)))
  if(max_difference < comp_value){
    return (TRUE)
  } else {
    return (FALSE)
  }
}

# Apply the function to the first N seeds
N <- 10^1
results <- c(rep(0,N))
for(j in 1:N){
  range <- ((j-1)*1000+1):((j-1)*1000+1000)
  results[j] <- sum(apply(mapply(generator, seed=range), MARGIN = 2, FUN = kolmogorov))
}

# Results
sum(results[results>=900])/length(results)
hist(results, breaks = c(980:1000), freq = FALSE)
```