Monte Carlo Simulation

Ananda Biswas

2023-06-16

```r
give_inputs <- function(){
  # n = as.numeric(readline(prompt = "Enter the number of random numbers you want to generate = "))

  n = as.integer(params$number_of_random_numbers)

  # our_seed = as.numeric(readline(prompt = "Enter seed = "))

  our_seed = as.integer(params$seed)

  # a = as.numeric(readline(prompt = "Enter the value of the constant a = "))

  a = as.integer(params$constant_a)

  # b = as.numeric(readline(prompt = "Enter the value of the constant b = "))

  b = as.integer(params$constant_b)

  # m = as.numeric(readline(prompt = "Enter the value of the constant m = "))

  m = as.integer(params$constant_m)

  our_inputs <- c(n, our_seed, a, b, m)

  return(our_inputs)
}
```

```r
give_random_numbers <- function(inputs){

  n = inputs[1]

  seed_value = inputs[2]

  a = inputs[3]

  b = inputs[4]

  m = inputs[5]

  numbers <- c(seed_value)

  for (i in 2:(n+1)) {
    numbers <- append(numbers, (a * numbers[i-1] + b) %% m, after = length(numbers))
  }
  return(numbers[-1])
}
```

Linear Congruence Method

Here we use the recursive relation :

$$x_n = ax_{n-1} + b \text{ (mod m)}$$

where x_0 is the seed value provided by the user, a, b and m are constants that can also be provided by the user.

```
my_inputs <- give_inputs()

our_random_numbers <- give_random_numbers(my_inputs)

print(our_random_numbers)
```

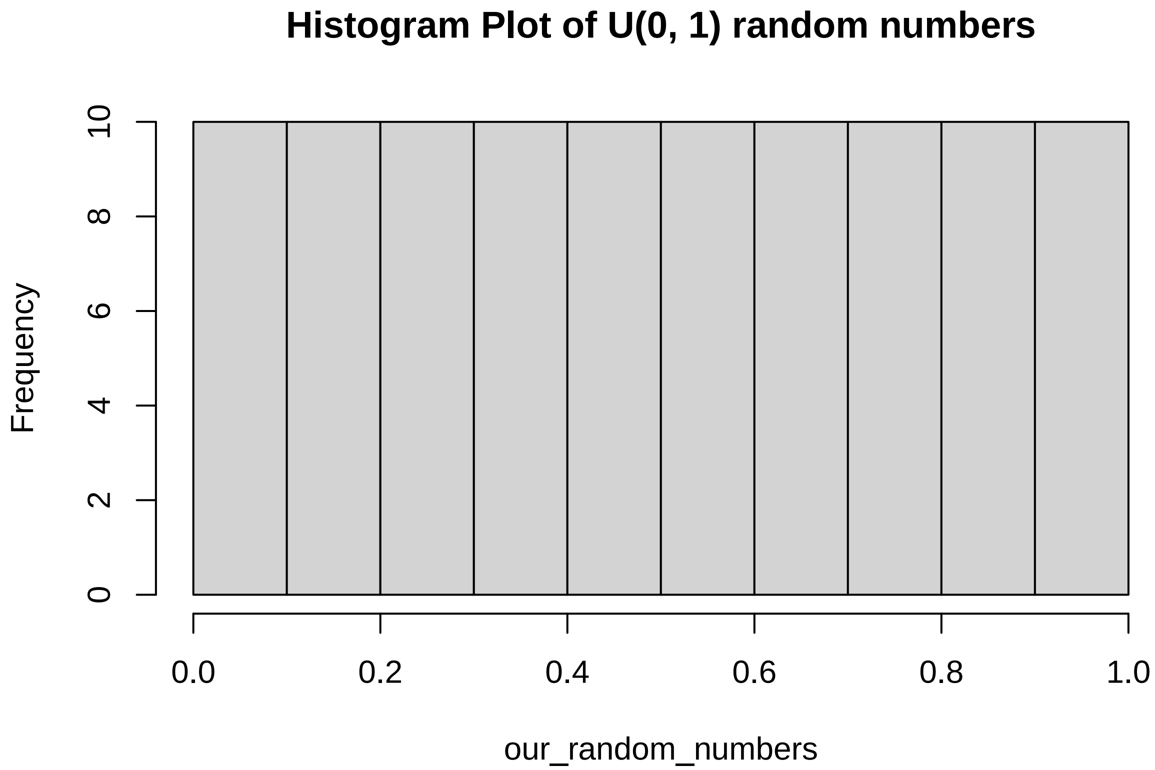
```
##     [1] 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31 82 53 44 55 86 37  8 99
##    [25] 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75  6 57 28 19 30 61 12 83
##    [49] 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39 50 81 32  3 94  5 36 87
##    [73] 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23 14 25 56  7 78 69 80 11
##    [97] 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31 82 53 44 55
##   [121] 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75  6 57 28 19
##   [145] 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39 50 81 32  3
##   [169] 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23 14 25 56  7
##   [193] 78 69 80 11 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31
##   [217] 82 53 44 55 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75
##   [241]  6 57 28 19 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39
##   [265] 50 81 32  3 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23
##   [289] 14 25 56  7 78 69 80 11 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27
##   [313] 98 89  0 31 82 53 44 55 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51
##   [337]  2 73 64 75  6 57 28 19 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95
##   [361] 26 77 48 39 50 81 32  3 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59
##   [385] 70  1 52 23 14 25 56  7 78 69 80 11 62 33 24 35 66 17 88 79 90 21 72 43
##   [409] 34 45 76 27 98 89  0 31 82 53 44 55 86 37  8 99 10 41 92 63 54 65 96 47
##   [433] 18  9 20 51  2 73 64 75  6 57 28 19 30 61 12 83 74 85 16 67 38 29 40 71
##   [457] 22 93 84 95 26 77 48 39 50 81 32  3 94  5 36 87 58 49 60 91 42 13  4 15
##   [481] 46 97 68 59 70  1 52 23 14 25 56  7 78 69 80 11 62 33 24 35 66 17 88 79
##   [505] 90 21 72 43 34 45 76 27 98 89  0 31 82 53 44 55 86 37  8 99 10 41 92 63
##   [529] 54 65 96 47 18  9 20 51  2 73 64 75  6 57 28 19 30 61 12 83 74 85 16 67
##   [553] 38 29 40 71 22 93 84 95 26 77 48 39 50 81 32  3 94  5 36 87 58 49 60 91
##   [577] 42 13  4 15 46 97 68 59 70  1 52 23 14 25 56  7 78 69 80 11 62 33 24 35
##   [601] 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31 82 53 44 55 86 37  8 99
##   [625] 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75  6 57 28 19 30 61 12 83
##   [649] 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39 50 81 32  3 94  5 36 87
##   [673] 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23 14 25 56  7 78 69 80 11
##   [697] 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31 82 53 44 55
##   [721] 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75  6 57 28 19
##   [745] 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39 50 81 32  3
##   [769] 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23 14 25 56  7
##   [793] 78 69 80 11 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27 98 89  0 31
##   [817] 82 53 44 55 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51  2 73 64 75
##  [841]  6 57 28 19 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95 26 77 48 39
##  [865] 50 81 32  3 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59 70  1 52 23
##  [889] 14 25 56  7 78 69 80 11 62 33 24 35 66 17 88 79 90 21 72 43 34 45 76 27
##  [913] 98 89  0 31 82 53 44 55 86 37  8 99 10 41 92 63 54 65 96 47 18  9 20 51
##  [937]  2 73 64 75  6 57 28 19 30 61 12 83 74 85 16 67 38 29 40 71 22 93 84 95
##  [961] 26 77 48 39 50 81 32  3 94  5 36 87 58 49 60 91 42 13  4 15 46 97 68 59
##  [985] 70  1 52 23 14 25 56  7 78 69 80 11 62 33 24 35
```

Generating $U(0, 1)$ Random Numbers

```r
my_inputs <- give_inputs()

my_inputs[1] <- as.integer(params$number_of_uniform_0_1_random_numbers)

our_random_numbers <- give_random_numbers(my_inputs)

our_random_numbers <- our_random_numbers / max(our_random_numbers)

print(our_random_numbers)
```

```
##   [1] 0.66666667 0.17171717 0.88888889 0.79797980 0.90909091 0.21212121
##   [7] 0.72727273 0.43434343 0.34343434 0.45454545 0.76767677 0.27272727
##  [13] 0.98989899 0.89898990 0.00000000 0.31313131 0.82828283 0.53535354
##  [19] 0.44444444 0.55555556 0.86868687 0.37373737 0.08080808 1.00000000
##  [25] 0.10101010 0.41414141 0.92929293 0.63636364 0.54545455 0.65656566
##  [31] 0.96969697 0.47474747 0.18181818 0.09090909 0.20202020 0.51515152
##  [37] 0.02020202 0.73737374 0.64646465 0.75757576 0.06060606 0.57575758
##  [43] 0.28282828 0.19191919 0.30303030 0.61616162 0.12121212 0.83838384
##  [49] 0.74747475 0.85858586 0.16161616 0.67676768 0.38383838 0.29292929
##  [55] 0.40404040 0.71717172 0.22222222 0.93939394 0.84848485 0.95959596
##  [61] 0.26262626 0.77777778 0.48484848 0.39393939 0.50505051 0.81818182
##  [67] 0.32323232 0.03030303 0.94949495 0.05050505 0.36363636 0.87878788
##  [73] 0.58585859 0.49494949 0.60606061 0.91919192 0.42424242 0.13131313
##  [79] 0.04040404 0.15151515 0.46464646 0.97979798 0.68686869 0.59595960
##  [85] 0.70707071 0.01010101 0.52525253 0.23232323 0.14141414 0.25252525
##  [91] 0.56565657 0.07070707 0.78787879 0.69696970 0.80808081 0.11111111
##  [97] 0.62626263 0.33333333 0.24242424 0.35353535
```

```r
hist(our_random_numbers, main = "Histogram Plot of U(0, 1) random numbers")
```



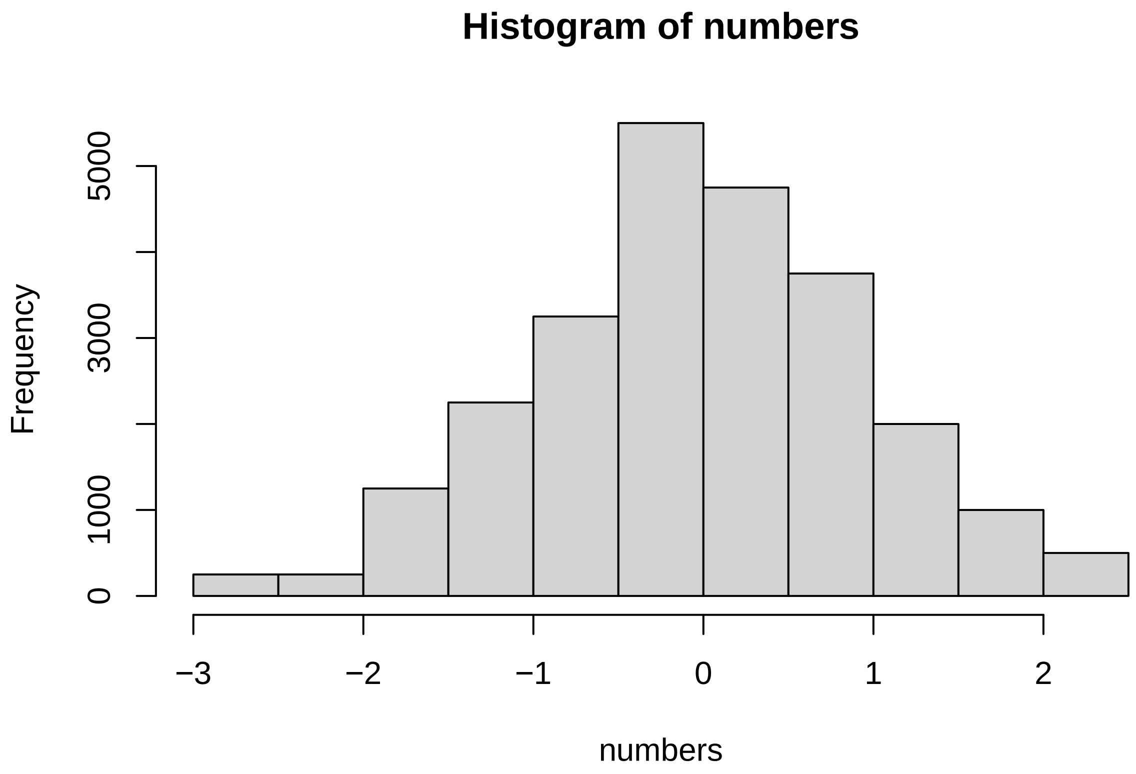
Generating $U(a, b)$ Random Numbers using CDF Inversion Method

```
numbers_1 <- c(73)

n = 25000
a = 21
b = 31
m = 100

for (i in 2:(n+1)) {
    numbers_1 <- append(numbers_1, (a * numbers_1[i-1] + b) %% m, after = length(numbers_1))
}

numbers_1 <- numbers_1 / max(numbers_1)

numbers_1 <- numbers_1[-1]

numbers_2 <- c(89)

for (i in 2:(n+1)) {
    numbers_2 <- append(numbers_2, (a * numbers_2[i-1] + b) %% m, after = length(numbers_2))
}

numbers_2 <- numbers_2 / max(numbers_2)
```

```
numbers_2 <- numbers_2[-1]
```

```
standard_normal_numbers <- c()

for (i in 1:length(numbers_1)) {
  temp <- sqrt((-2) * log(numbers_1[i])) * sin(2 * pi * numbers_2[i])

  standard_normal_numbers <- append(standard_normal_numbers, temp, after = length(standard_normal_numbe
}
```

```
numbers <- standard_normal_numbers
hist(numbers)
```


Histogram of numbers
Frequency
numbers