Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Teacher: Fred Torres Cruz
Student: Anderson Brian Flores Suaña

`REPOSITORY: https://github.com/andersonfloress/Trabajos_Est_Computacional`

HOMEWORK 05

Introduction

The code implements a linear congruential generator (LCG) in the R programming language and performs a detailed comparison with R's built-in random number generators. These types of generators are fundamental in simulations, cryptography, sampling, and other areas where seemingly random values are required. Below are the functions used and the analyses performed.

1. Functions for Pseudorandom Number Generation

`lcg_core(n, seed, a, c, m)`
This function is the core of the linear congruential generator. It uses a classic mathematical formula to generate sequences of pseudorandom numbers:

$$X_{n+1} = (a \times X_n + c) \quad \text{mód } m$$

Where:

- `n` is the number of numbers to generate.
- `seed` is the initial seed, which ensures reproducibility of results.
- `a` is the multiplicative coefficient.
- `c` is the increment (additive term).
- `m` is the modulus (defines the range of possible values).

The function returns a vector of integer values ranging from 0 to m - 1. Although simple, this type of generator can be very fast, but it requires a good choice of parameters to produce well-distributed sequences with long periods.
`lcg_real(n, min_val, max_val, seed, a, c, m)`
This function takes the output from `lcg_core` and transforms it into real numbers uniformly distributed within the interval [min_val, max_val]. It normalizes by dividing by `m`,

then scales to the desired range. By default, it generates values between 0 and 1. This conversion is useful for simulating continuous variables.
`lcg_int(n, min_val, max_val, seed, a, c, m)`
Unlike `lcg_real`, this function transforms the values into integers within a specific interval. It uses the `floor` function to round down, and `pmin` to ensure the maximum value is not exceeded. This method is useful for simulating discrete events, such as dice rolls or random choices with integer values.

2. Comparisons Made

The code evaluates the quality of the manually implemented LCG generator by comparing it to R's native generators (`runif` and `sample`). These comparisons include both visual and statistical analyses.
Comparison of Random Real Numbers
A total of 1000 real numbers in the interval [0, 1] are generated using:

- `runif`: R's native generator, which provides good statistical quality.
- `lcg_real`: the custom generator based on the LCG method.

Histograms are created to visualize the data distribution and a set of descriptive statistics is calculated:

- Mean
- Variance
- Standard deviation
- Minimum and maximum
- Range

This comparison helps to observe whether the LCG generator distributes the values uniformly or if there are undesirable patterns or biases.
Comparison of Random Integer Numbers
A series of die rolls is simulated (values between 1 and 6). 1000 values are generated using:

- `sample`: R's native random integer generator with replacement.
- `lcg_int`: custom integer generator.

Again, both data sets are statistically compared to assess the uniformity and accuracy of the LCG generator when simulating discrete events.

The code provides a clear and practical way to compare a basic linear congruential generator with the high-quality random generators included in R. Through visual and statistical analyses, one can assess whether the custom LCG meets the minimum randomness requirements for possible use in simulations or other applications. Although congruential generators can be simpler and faster, modern generators like those in R tend to offer greater robustness and precision, so this type of comparison is key to making informed decisions about their use.

EVIDENCE:

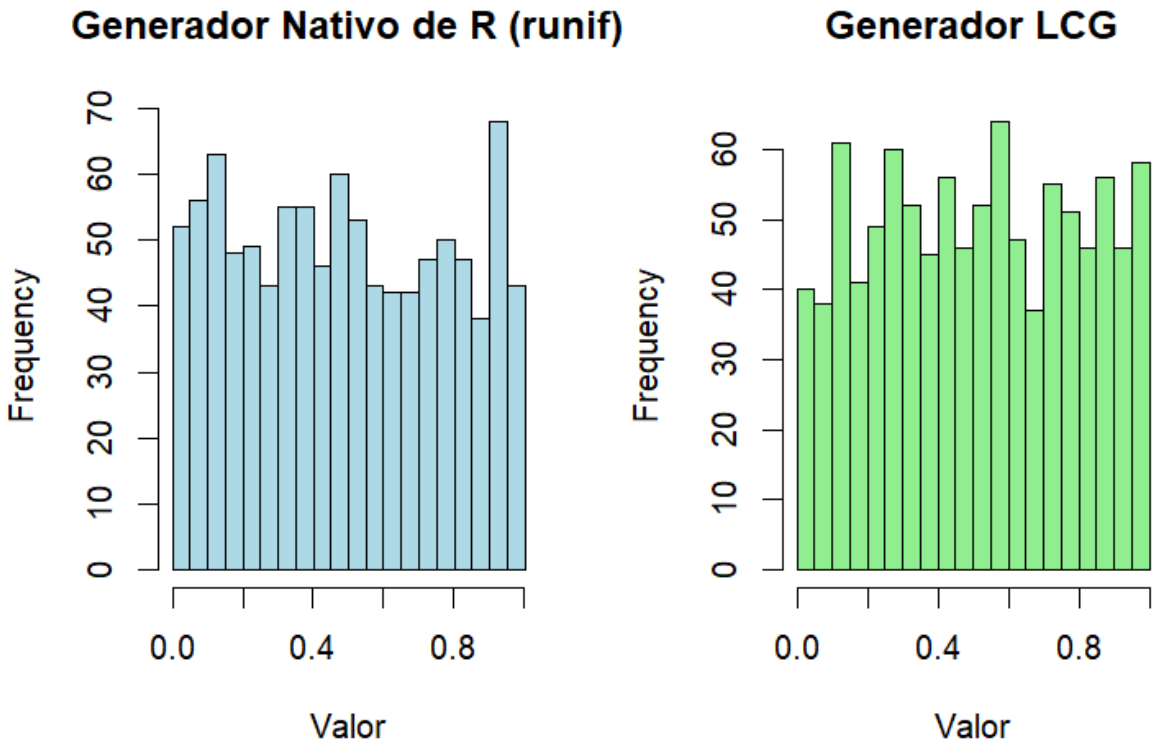
```
+ }
> #=====COMPARACIONES=====
> # Generar valores con LCG
> seed <- 42
> n <- 1000
> valores_lcg <- lcg_real(n, 0, 1, seed = seed)
> set.seed(seed)
> valores_r <- runif(n)
> # Comparacion visual
> par(mfrow = c(1, 2))
> hist(valores_r, breaks = 30, col = "lightblue", main = "Generador Nativo de R (runif)", xlab = "Valor")
> hist(valores_lcg, breaks = 30, col = "lightgreen", main = "Generador LCG", xlab = "Valor")
> # Comparaciones estadisticas
> cat("\n--- Comparación con Generador Nativo de R ---\n")

--- Comparación con Generador Nativo de R ---
> cat("Media (R nativo):", mean(valores_r), "\n")
Media (R nativo): 0.4882555
> cat("Media (LCG):", mean(valores_lcg), "\n")
Media (LCG): 0.5104797
> cat("Varianza (R nativo):", var(valores_r), "\n")
Varianza (R nativo): 0.08493159
> cat("Varianza (LCG):", var(valores_lcg), "\n")
Varianza (LCG): 0.08148821
> cat("Desviación estándar (R nativo):", sd(valores_r), "\n")
Desviación estándar (R nativo): 0.2914302
> cat("Desviación estándar (LCG):", sd(valores_lcg), "\n")
Desviación estándar (LCG): 0.2854614
> cat("Mínimo (R nativo):", min(valores_r), "\n")
```

```
> cat("Mínimo (R nativo):", min(valores_r), "\n")
Mínimo (R nativo): 0.0002388966
> cat("Mínimo (LCG):", min(valores_lcg), "\n")
Mínimo (LCG): 9.778887e-09
> cat("Máximo (R nativo):", max(valores_r), "\n")
Máximo (R nativo): 0.9984908
> cat("Máximo (LCG):", max(valores_lcg), "\n")
Máximo (LCG): 0.9993122
> cat("Rango (R nativo):", range(valores_r), "\n")
Rango (R nativo): 0.0002388966 0.9984908
> cat("Rango (LCG):", range(valores_lcg), "\n")
Rango (LCG): 9.778887e-09 0.9993122
> cat("\n--- Comparación: lcg_int vs sample() ---\n")

--- Comparación: lcg_int vs sample() ---
> set.seed(42)
> valores_r_int <- sample(1:6, 1000, replace = TRUE)
> valores_lcg_int <- lcg_int(1000, 1, 6, seed = 42)
> cat("Media (R nativo):", mean(valores_r_int), "\n")
Media (R nativo): 3.413
> cat("Media (LCG):", mean(valores_lcg_int), "\n")
Media (LCG): 3.559
> cat("Varianza (R nativo):", var(valores_r_int), "\n")
Varianza (R nativo): 2.93937
> cat("Varianza (LCG):", var(valores_lcg_int), "\n")
Varianza (LCG): 2.895414
```

```
> cat("Varianza (R nativo):", var(valores_r_int), "\n")
Varianza (R nativo): 2.93937
> cat("Varianza (LCG):", var(valores_lcg_int), "\n")
Varianza (LCG): 2.895414
> cat("Desviación estándar (R nativo):", sd(valores_r_int), "\n")
Desviación estándar (R nativo): 1.714459
> cat("Desviación estándar (LCG):", sd(valores_lcg_int), "\n")
Desviación estándar (LCG): 1.701592
> cat("Mínimo (R nativo):", min(valores_r_int), "\n")
Mínimo (R nativo): 1
> cat("Mínimo (LCG):", min(valores_lcg_int), "\n")
Mínimo (LCG): 1
> cat("Máximo (R nativo):", max(valores_r_int), "\n")
Máximo (R nativo): 6
> cat("Máximo (LCG):", max(valores_lcg_int), "\n")
Máximo (LCG): 6
> cat("Rango (R nativo):", range(valores_r_int), "\n")
Rango (R nativo): 1 6
> cat("Rango (LCG):", range(valores_lcg_int), "\n")
Rango (LCG): 1 6
>
```


Generador Nativo de R (runif)
Frequency
0
10
20
30
40
50
60
70
0.0
0.4
0.8
Valor
Generador LCG
Frequency
0
10
20
30
40
50
60
0.0
0.4
0.8
Valor