UNIVERSIDAD NACIONAL DE LOJA

PERIODO ACADEMICO: OCTUBRE 2019 – MARZO 2020
PRACTICA # 5
ASIGNATURA: SIMULACIÓN

RESULTADO DE APRENDIZAJE DE LA PRÁCTICA: Entiende los métodos de generación de números uniformemente distribuidos usando R y obtiene el valor esperado usando la simulación Montecarlo

TIEMPO PLANIFICADO: 3 HORAS
NUMERO DE ESTUDIANTES: Sexto ciclo (Paralelo A)

1. TEMA: Generación de números aleatorios y Simulación Montecarlo

2. OBJETIVOS:

- Comprende los algoritmos y funciones en R para la generación de números aleatorios.
- Comprende la simulación Monte Carlo para la obtención del valor esperado.

3. RECURSOS NECESARIOS:

- R
- Computador de Laboratorios

4. INSTRUCCIONES:

- Prohibido consumo de alimentos
- Prohibido equipo de diversión, celulares etc.
- Prohibido jugar
- Prohibido mover o intercambiar los equipos de los bancos de trabajo
- Prohibido sacar los equipos del laboratorio sin autorización.
- Ubicar los equipos y accesorios en el lugar dispuesto por el responsable del laboratorio, luego de terminar las prácticas.
- Uso adecuado de equipos

5. ACTIVIDADES POR DESARROLLAR:

1. Generate 20 pseudorandom numbers using $x_n = 172\ x_{n-1} \pmod{30\ 307}$, with initial seed $x_0 = 17\ 218$.

```
random.number = numeric(20)
random.seed = 17218
for (i in 1:20) {
        random.seed = (172*random.seed) %% 30307
        random.number[i] = random.seed / 30307
}
random.number
```

```
 [1] 0.71656713 0.24954631 0.92196522 0.57801828 0.41914409 0.09278385 0.95882139
 [8] 0.91727984 0.77213185 0.80667833 0.74867192 0.77157092 0.71019896 0.15422180
[15] 0.52614907 0.49764081 0.59421916 0.20569505 0.37954928 0.28247600
```

2. Generate 20 pseudorandom numbers using the multiplicative congruential generator with $b = 171$ and $m = 32\,767$ with an initial seed of 2019.

```
random.number = numeric(20)
random.seed = 2019
for (i in 1:20) {
        random.seed = (171 * random.seed) %% 32767
        random.number[i] = random.seed / 32767
}
random.number
```

```
 [1] 0.53648488 0.73891415 0.35431990 0.58870205 0.66805017 0.23657949 0.45509201
 [8] 0.82073428 0.34556108 0.09094516 0.55162206 0.32737205 0.98062075 0.68614765
[15] 0.33124790 0.64339122 0.01989807 0.40256966 0.83941160 0.53938414
```

3. Use the `runif()` function (with `set.seed(32078)`) to generate 10 pseudorandom numbers from (a) the uniform (0, 1) distribution

```
set.seed(32078)
runif(10)
```

```
[1] 0.2564626 0.4988177 0.5266549 0.6269816 0.8052754 0.1843452 0.5102327 0.3683905
[9] 0.1708176 0.7432888
```

(b) (the uniform (3, 7) distribution

```
set.seed(32078)
runif(10, min = 3, max = 7)
```

```
 [1] 4.025850 4.995271 5.106620 5.507927 6.221102 3.737381 5.040931 4.473562 3.683270
[10] 5.973155
```

(c) the uniform (−2, 2) distribution.

```
set.seed(32078)
runif(10, min = -2, max = 2)ç
```

```
[1] -0.974149697 -0.004729333  0.106619657  0.507926506  1.221101642 -1.262619189
[7]  0.040930690 -0.526437979 -1.316729628  0.973155177
```

4. Generate 1000 uniform pseudorandom variates using the `runif()` function, assigning them to a vector called `U`. Use `set.seed(19908)`.

(a) Compute the average, variance, and standard deviation of the numbers in `U`.

```
set.seed(19908)
U <-c(runif(1000))
resultado<- c(mean(U),var(U),sd(U))
resultado
```

```
[1] 0.49605698 0.08148008 0.28544716
```

(b) Compare your results with the true mean, variance, and standard deviation.

```
media<-0.5
varianza <- 1/12
desv_estan<-sqrt(varianza)
comp <-c(media,varianza,desv_estan)
comp
resultado
```

```
[1] 0.50000000 0.08333333 0.28867513
> resultado
[1] 0.49605698 0.08148008 0.28544716
```

(c) Compute the proportion of the values of `U` that are less than 0.6, and compare with the probability that a uniform random variable U is less than 0.6.

```
sum(U<0.6)/1000
punif(0.6)
```

```
[1] 0.61
[1] 0.6
```

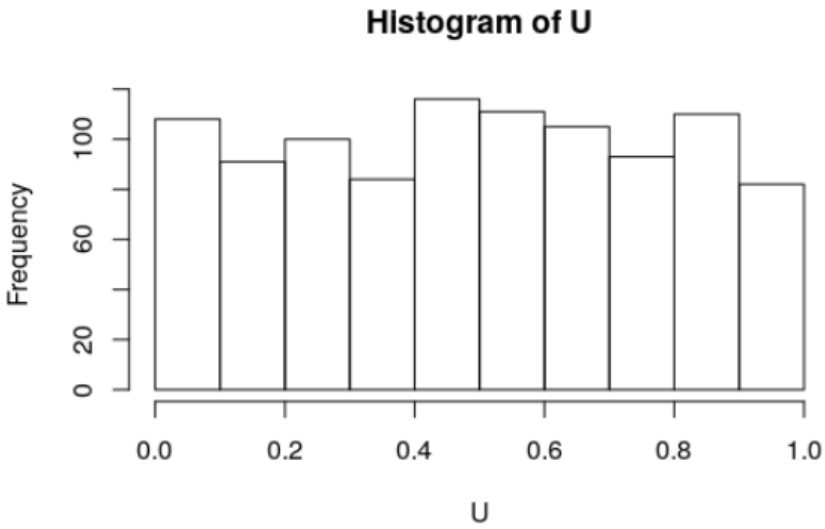
(d) Estimate the expected value of $1/(U + 1)$.

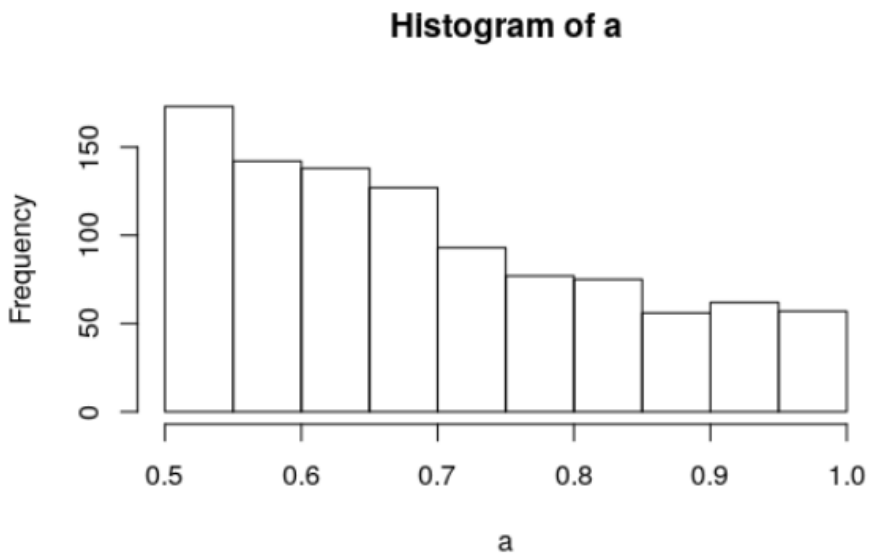
```
a <- 1/(U + 1)
mean(a)
```

```
[1] 0.6946063
```

(e) Construct a histogram of the values of `U`, and of `1/(U+1)`.

```
hist(U)
hist(a)
```





5. Simulate 10 000 independent observations on a uniformly distributed random variable on the interval [3.7,5.8].

   (a) Estimate the mean, variance, and standard deviation of such a uni- form random variable and compare your estimates with the true values.

```
r<-c(runif(10000,min=3.7,max=5.8))
r
mteorica <-(3.7+5.8)/2
vteorica <- (5.8-3.7)^2/12
dteorica <- sqrt((5.8-3.7)^2/12)
estimacion <-c(mean(r),var(r),sd(r))
V_real <- c(mteorica,vteorica,dteorica)
estimacion
V_real
```

```
> estimacion
[1] 4.7448701 0.3648599 0.6040364
>             V_real
[1] 4.7500000 0.3675000 0.6062178
```

   (b) Estimate the probability that such a random variable is greater than 4.0. Compare with the true value.

```
probabilidad<- length(r[r>4])/length(r)
probabilidad
```

```
[1] 0.8575
```

6. Simulate 10 000 values of a uniform (0, 1) random variable, $U1$, using runif(), and simulate another set of 10 000 values of a uniform (0, 1) random variable $U2$. Assign these vectors to U1 and U2, respectively. Since the values in U1 and U2 are approximately independent, we can view $U1$ and $U2$ as independent uniform (0, 1) random variables.

   (a) Estimate $E[U1 + U2]$. Compare with the true value, and compare with an estimate of $E[U1] + E[U2]$.

```
E[U1]+E[U2]
U1 <- runif(10000)
```

```
U2 <- runif(10000)
U <- U1+U2
media= mean(U)
media
med <- mean(U1)+mean(U2)
med

[1] 0.9999159
[1] 0.9999159
```

(b) Estimate Var($U1 + U2$) and Var($U1$) + Var($U2$). Are they equal? Should the true values be equal?.

```
varianza<-var(U)
varianza
vari <- var(U1)+var(U2)
vari

[1] 0.1626053
[1] 0.16289
```

(c) Estimate $P(U1 + U2 \leq 1.5)$.

```
prob <- length(U[U <= 1.5])/length(U)
prob

[1] 0.8792
```

(d) Estimate $P(\sqrt{U1} + \sqrt{U2}) \leq 1.5)$

```
V <- sqrt(U1)+sqrt(U2)
pro <- length(V[V <= 1.5])/length(V)
pro

[1] 0.6562
```

7. Suppose U1, U2 and U3 are independent uniform random variables on the interval (0, 1). Use simulation to estimate the following quantities:

(a) E[U1 + U2 + U3].

```
U1<- runif(15)
U2<- runif(15)
U3<- runif(15)
suma <-U1+U2+U3
media <-mean(suma)
media

[1] 1.607105
```

(b) Var(U1 + U2 + U3) and Var(U1) + Var(U2) + Var(U3).

```
varianza <-var(suma)
varianza
```

[1] 0.1792929

```
SumVarianza  <- var(U1)+var(U2)+var(U3)
SumVarianza
```

[1] 0.1994154

(c) E[√(U 1 + U 2 + U 3)]

```
raiz <- sqrt(suma)
mediaRaiz  <- mean(raiz)
mediaRaiz
```

[1] 1.25763

(d) P(√u 1 + √U 2 + √U 3 ≥ 0.8)

```
R <- sqrt(U1)+sqrt(U2)+  sqrt(U3)
#Provabilidad que sea mayor a 0.8
R
```

```
 [1] 2.094666 2.450485 2.075611 2.376935 1.619982 1.806570 1.973090 2.127924 1.873205
[10] 1.772588 2.600281 2.657026 2.313315 2.226839 1.608716
```

```
P<- length(R[R>=0.8])/length(R)
P
```

[1] 1

6. INVESTIGACIÓN COMPLEMENTARIA (a elaborar por el estudiante)

Investigar acerca de la Simulación Monte Carlo (usos, ventajas y desventajas)

La simulación de Montecarlo es un método estadístico utilizado para resolver problemas matemáticos complejos a través de la generación de variables aleatorias.
La simulación de Montecarlo o método de Montecarlo, le debe el nombre al famoso casino del principado de Mónaco. La ruleta es el juego de casino más famoso y también el ejemplo más sencillo de mecanismo que permite generar números aleatorios.
La clave de este método está en entender el término 'simulación'. Realizar una simulación consiste en repetir o duplicar las características y comportamientos de un sistema real. Así pues, el objetivo principal de la simulación de Montecarlo es intentar imitar el comportamiento de variables reales para, en la medida de lo posible, analizar o predecir cómo van a evolucionar.
A través de la simulación se pueden resolver desde problemas muy sencillos, hasta problemas muy complejos. Algunos problemas pueden solucionarse con papel y bolígrafo. Sin embargo, la mayoría requieren el uso de programas informáticos como Excel, R Studio o Matlab. Sin estos programas, resolver determinados problemas llevaría muchísimo tiempo.

Usos

En economía la simulación de Montecarlo se utiliza tanto en empresas como en inversión. Siendo en el mundo de la inversión donde más se utiliza. Algunos ejemplos de simulación de Montecarlo en inversión son los siguientes:

- Crear, valorar y analizar carteras de inversión
- Valorar productos financieros complejos como las opciones financieras
- Creación de modelos de gestión de riesgo

Dado que la rentabilidad de una inversión es impredecible se utiliza este tipo de método para evaluar

distintos tipos de escenarios. Un ejemplo sencillo se encuentra en la bolsa de valores .Los movimientos de una acción no se pueden predecir. Se pueden estimar, pero es imposible hacerlo con exactitud. Por ello, mediante la simulación de Montecarlo, se intenta imitar el comportamiento de una acción o de un conjunto de ellas para analizar cómo podrían evolucionar.
Una vez se realiza la simulación de Montecarlo se extraen una cantidad muy grande de escenarios posibles.

Ventajas

- Es un método directo y flexible.
- Existe un amplio abanico de programas y lenguajes destinados a simular.
- Cuando el modelo matemático es demasiado complicado la simulación permite obtener una aproximación.
- La simulación nos permite formular condiciones extremas con riesgos nulos.
- La simulación no interfiere con el mundo real. Permite experimentar.
- Permite estudiar la interacción entre las diferentes variables del problema.
- Mediante la simulación podemos "influir en el tiempo" de los procesos.
- La simulación permite resolver problemas que no tienen solución analítica.

Desventajas

- Una buena simulación puede resultar muy complicada, gran número de variables.
- La simulación no genera soluciones Optimas globales.
- No proporciona la decisión a tomar, sino que resuelve el problema mediante aproximación para unas condiciones iniciales.
- Cada simulación es única, interviene el azar.

7. DISCUSIÓN (a elaborar por el estudiante)

Los ejercicios planteados permiten enriquecer el conocimiento de cómo se debe utilizar R Studio, del mismo modo permiten conocer funciones propias de este lenguaje para generar números pseudoaleatorios.

8. CONCLUSIONES (a elaborar por el estudiante)

R Studio presenta múltiples funciones propias del programa que ahorran largas líneas de código, como la función var(), que permite calcular varianzas, la función mean() que calcula el promedio de una cantidad de número, entre otras funciones útiles que otros lenguajes de programación no dispones.

9. RECOMENDACIONES (a elaborar por estudiante)

Es necesario tener en claro las fórmulas que se necesitan para generar números psudoaleatoiros, en este caso para la actividad, las formulas del congruencial mixto y multiplicativo.

10. BIBLIOGRAFÍA:

- Michael Luby, *Pseudorandomness and Cryptographic Applications*, Princeton Univ Press, 1996. A definitive source of techniques for provably random sequences.
- Donald Knuth. *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89684-2. Chapter 3, pp. 1-193. Extensive coverage of statistical tests for non-randomness.
- R. Matthews *Maximally Periodic Reciprocals* Bulletin of the Institute of Mathematics and its Applications 28 147-148 1992

Firma del Presidente de Curso de Sexto A

Ing. Marlon Santiago Viñan
Ludeña Mg. Sc DOCENTE CIS