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**
   Link del Archivo. R:
   https://github.com/DeibyCalva/Practica_5_en_R/blob/master/practica_5DeibyCalva.R

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)# generar 20 numeros
random.seed = 17218 #semilla inicial x0 = 17218.
for (j in 1:20) {
  random.seed = (172*random.seed)%%30307
  random.number[j] = random.seed/30307
}
random.number
```

```
> random.number
 [1] 0.71656713 0.24954631 0.92196522 0.57801828 0.41914409
 [6] 0.09278385 0.95882139 0.91727984 0.77213185 0.80667833
[11] 0.74867192 0.77157092 0.71019896 0.15422180 0.52614907
[16] 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 #semilla
for (j in 1:20) {
  random.seed = (171*random.seed)%%32767
  random.number[j] = random.seed/32767
}
random.number
```

```
> random.number
 [1] 0.53648488 0.73891415 0.35431990 0.58870205 0.66805017
 [6] 0.23657949 0.45509201 0.82073428 0.34556108 0.09094516
[11] 0.55162206 0.32737205 0.98062075 0.68614765 0.33124790
[16] 0.64339122 0.01989807 0.40256966 0.83941160 0.53938414
```

3. Use the `runif()` function (with `set.seed(32078)`)to generate10 pseudorandom numbers from
(a) the uniform (0, 1) distribution

(b) the uniform (3, 7) distribution

(c) the uniform (−2, 2) distribution.

```
##opcion a : la distribución uniforme (0,1)
set.seed(32078)
runif(10, min = 0,max = 1)
##opcion b :  la distribución uniforme (3, 7)
set.seed(32078)
runif(10, min = 3,max = 7)
##opcion c :  la distribución uniforme (-2, 2).
set.seed(32078)
runif(10, min = -2,max = 2)
```

```
> ##opcion a : la distribución uniforme (0,1)
> set.seed(32078)
> runif(10, min = 0,max = 1)
 [1] 0.2564626 0.4988177 0.5266549 0.6269816 0.8052754 0.1843452
 [7] 0.5102327 0.3683905 0.1708176 0.7432888
> ##opcion b :  la distribución uniforme (3, 7)
> set.seed(32078)
> runif(10, min = 3,max = 7)
 [1] 4.025850 4.995271 5.106620 5.507927 6.221102 3.737381 5.040931
 [8] 4.473562 3.683270 5.973155
> ##opcion c :  la distribución uniforme (-2, 2).
> set.seed(32078)
> runif(10, min = -2,max = 2)
 [1] -0.974149697 -0.004729333  0.106619657  0.507926506
 [5]  1.221101642 -1.262619189  0.040930690 -0.526437979
 [9] -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)`.

```
U <- numeric(1000)
set.seed(19908)
U<- runif(1000)
U
```

```
[913] 4.940431e-01 7.185894e-01 2.773835e-01 7.113573e-01
[917] 4.347484e-01 3.728047e-01 3.718406e-01 7.906081e-02
[921] 4.013646e-01 9.610867e-01 5.220548e-01 1.327206e-01
[925] 4.942681e-01 2.251840e-01 5.227789e-01 9.393339e-01
[929] 2.342448e-01 1.772925e-01 9.968368e-01 7.385375e-02
[933] 5.332675e-01 3.688998e-01 4.836063e-01 6.298468e-01
[937] 6.597873e-01 3.317279e-01 7.189077e-02 6.270664e-01
[941] 5.116087e-01 5.492110e-01 7.991381e-01 6.156600e-02
[945] 8.404876e-01 2.625669e-01 7.247664e-01 5.549284e-01
[949] 3.046334e-01 9.824533e-01 5.240208e-01 4.750005e-01
[953] 2.556153e-01 7.024066e-01 9.265536e-01 8.196241e-01
[957] 6.689298e-01 8.897807e-02 5.295467e-01 4.347422e-02
[961] 9.797459e-01 4.589016e-01 3.756425e-01 2.923428e-01
[965] 1.423046e-01 5.115689e-01 2.679716e-02 2.951454e-01
[969] 8.130115e-03 4.059647e-01 8.877038e-01 8.019279e-01
[973] 9.976425e-01 7.079758e-01 7.165677e-01 3.715258e-01
[977] 2.040730e-01 9.830226e-02 9.671385e-01 3.020341e-02
[981] 4.638514e-01 5.042366e-01 9.824107e-02 6.041063e-01
[985] 4.575619e-01 9.435473e-01 5.169749e-01 4.620665e-01
[989] 8.087254e-01 4.018922e-01 9.215289e-01 5.593001e-01
[993] 5.602987e-01 8.952967e-01 6.098838e-01 9.674095e-01
[997] 4.686091e-01 3.689650e-01 2.988730e-01 6.428278e-01
```

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

```
###media
promedio<-sum(U)/length(U)
promedio
###Varianza
varianza<- sum(((U-promedio)^2)/length(U))
varianza
###Desviacion estandar
desv_estandar = sqrt(varianza)
desv_estandar
```

```
> ###media
> promedio<-sum(U)/length(U)
> promedio
[1] 0.496057
> ###Varianza
> varianza<- sum(((U-promedio)^2)/length(U))
> varianza
[1] 0.0813986
> ###Desviacion estandar
> desv_estandar = sqrt(varianza)
> desv_estandar
[1] 0.2853044
```

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

```
###media
promedio<-sum(U)/length(U)
promedio
mean(U)
###Varianza
varianza<- sum(((U-promedio)^2)/length(U))
varianza
var(U)
###Desviacion estandar
desv_estandar = sqrt(varianza)
desv_estandar
sd(U)
```

```
> ###media
> promedio<-sum(U)/length(U)
> promedio
[1] 0.496057
> mean(U)
[1] 0.496057
> ###Varianza
> varianza<- sum(((U-promedio)^2)/length(U))
> varianza
[1] 0.0813986
> var(U)
[1] 0.08148008
> ###Desviacion estandar
> desv_estandar = sqrt(varianza)
> desv_estandar
[1] 0.2853044
> sd(U)
[1] 0.2854472
```

(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.

```
#proporcion
prop.table(table(U<0.6))
#probabilidad
numbers <- runif(U,min=0,max=0.6)
number<-sample(numbers, 1)
punif(number)
```

```
> #proporcion
> prop.table(table(U<0.6))

FALSE  TRUE
 0.39  0.61
> #probabilidad
> numbers <- runif(U,min=0,max=0.6)
> number<-sample(numbers, 1)
> punif(number)
[1] 0.03722347
```

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

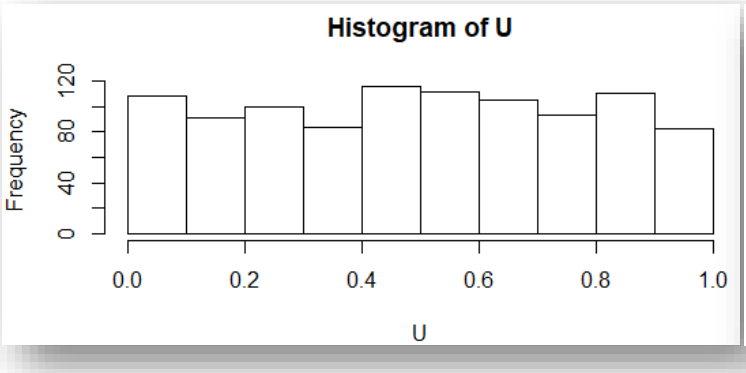
```
#opcion 6 calcular
mean(1/(U+1))
```

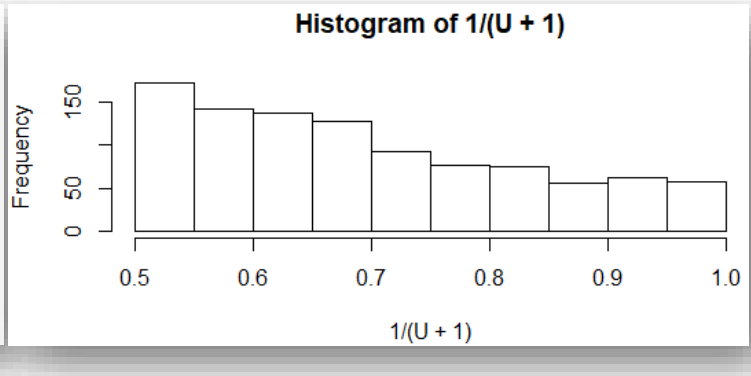
```
> mean(1/(U+1))
[1] 0.6946063
```

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

```
hist(U)

hist(1/(U+1))
```





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<-runif(10000,min=3.7,max=5.8)

# media
promedio<-sum(r)/length(r)
promedio
mean(r) # comparar con valor real
# Varianza
varianza<- sum(((r-promedio)^2)/length(r))
varianza
var(r)# comparar con valor real
# Desviacion estandar
desv_estandar = sqrt(varianza)
desv_estandar
sd(r)# comparar con valor real
```

```
> # media
> promedio<-sum(r)/length(r)
> promedio
[1] 4.74487
> mean(r) # comparar con valor real
[1] 4.74487
> # Varianza
> varianza<- sum(((r-promedio)^2)/length(r))
> varianza
[1] 0.3648234
> var(r)# comparar con valor real
[1] 0.3648599
> # Desviacion estandar
> desv_estandar = sqrt(varianza)
> desv_estandar
[1] 0.6040061
> sd(r)# comparar con valor real
[1] 0.6040364
```

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

```
length(r[r>4])/length(r)
punif(4, min = 3.7, max = 5.8, lower.tail = FALSE)
```

```
> length(r[r>4])/length(r)
[1] 0.8575
> punif(4, min = 3.7, max = 5.8, lower.tail = FALSE)
[1] 0.8571429
```

6. Simulate 10 000 values of a uniform $(0, 1)$ random variable, U_1, using `runif()`, and simulate another set of 10 000 values of a uniform $(0, 1)$ random variable U_2. Assign these vectors to `U1` and `U2`, respectively. Since the values in `U1` and `U2` are approximately independent, we can view U_1 and U_2 as independent uniform $(0, 1)$ random variables.

(a) Estimate $E[U_1 + U_2]$. Compare with the true value, and compare with an estimate of $E[U_1] + E[U_2]$.

```
# opcion a: Estimación E[U1 + U2]. Compare con el valor real y
#compare con una estimación de E[U1] + E[U2].
u1 <- runif(10000, min = 0, max = 1)
u2 <- runif(10000, min = 0, max = 1)
mean(u1+u2)
mean(u1)+mean(u2)
```

```
> mean(u1+u2)
[1] 0.9983046
> mean(u1)+mean(u2)
[1] 0.9983046
```

(b) Estimate $\text{Var}(U_1 + U_2)$ and $\text{Var}(U_1) + \text{Var}(U_2)$. Are they equal? Should the true values be equal?.

```
# opcion b: Estimación de Var(U1 + U2) y Var(U1) + Var(U2).
#¿Son iguales? Si los verdaderos valores sean iguales?.
var(u1+u2)
var(u1) + var(u2)
```

```
> var(u1+u2)
[1] 0.1632173
> var(u1) + var(u2)
[1] 0.164777
```

(c) Estimate $P(U_1 + U_2 \leq 1.5)$.

```
# opcion c: Estimación P(U1 + U2 ≤ 1.5).
mean(u1+u2)
mean(u1+u2) <= 1.5
```

```
> mean(u1+u2)
[1] 0.9983046
> mean(u1+u2) <= 1.5
[1] TRUE
```

(d) Estimate $P(\sqrt{U_1} + \sqrt{U_2}) \leq 1.5)$

```
# opcion d:
mean(sqrt(u1)+sqrt(u2))
mean(sqrt(u1)+sqrt(u2)) <= 1.5
```

```
> mean(sqrt(u1)+sqrt(u2))
[1] 1.333559
> mean(sqrt(u1)+sqrt(u2)) <= 1.5
[1] TRUE
```

7. Suppose U_1, U_2 and U_3 are independent uniform random variables on the interval $(0, 1)$. Use simulation to estimate the following quantities:

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

```
# opcion a
u1 <- runif(10000, min = 0, max = 1)
u2 <- runif(10000, min = 0, max = 1)
u3 <- runif(10000, min = 0, max = 1)
mean(u1+u2+u3)
```

```
> # opcion a
> u1 <- runif(10000, min = 0, max = 1)
> u2 <- runif(10000, min = 0, max = 1)
> u3 <- runif(10000, min = 0, max = 1)
> mean(u1+u2+u3)
[1] 1.496097
```

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

```
# opcion b
var(u1+u2+u3)
var(u1) + var(u2) + var(u3)
```

```
> # opcion b
> var(u1+u2+u3)
[1] 0.2522361
> var(u1) + var(u2) + var(u3)
[1] 0.2501159
```

(c) $E[\sqrt{U_1 + U_2 + U_3}]$

```
# opcion c
mean(sqrt(u1+u2+u3))
```

```
> # opcion c
> mean(sqrt(u1+u2+u3))
[1] 1.203789
```

(d) $P[\sqrt{U_1} + \sqrt{U_2} + \sqrt{U_3} \geq 0.8]$

```
#opcion d
mean(sqrt(u1)+sqrt(u2)+sqrt(u3))
mean(sqrt(u1)+sqrt(u2)+sqrt(u3)) >= 0.8
```

```
> #opcion d
> mean(sqrt(u1)+sqrt(u2)+sqrt(u3))
[1] 1.996441
> mean(sqrt(u1)+sqrt(u2)+sqrt(u3)) >= 0.8
[1] TRUE
>
```

6 INVESTIGACIÓN COMPLEMENTARIA (a elaborar por el estudiante)

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

Simulación Monte Carlo

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.

¿Para qué se utiliza la simulación de Montecarlo?

Claro que, lo importante es saber para qué se utiliza este método. Es decir, casos concretos para entender la importancia del método. 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:

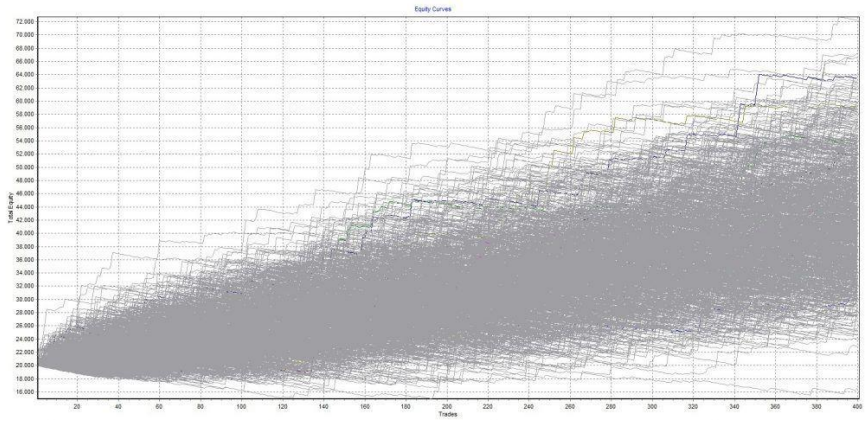
- Crear, valorar y analizar carteras de inversión
- Valorar productos financieros complejos como las opciones financieras
- Creación de modelos de gestión de riesgo

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.

Ejemplo de la simulación de Montecarlo

Supongamos que queremos contratar a un gestor que realice operaciones por nosotros en la bolsa de valores. El gestor presume de haber ganado 50% de rentabilidad durante el último año con una cuenta de valores de 20.000 dólares. Para confirmar que lo que dice es verdad, le pedimos su track record auditado. Es decir, el registro de todas sus operaciones verificado por una auditor (para evitar estafas y cuentas falsas). El gestor nos facilita toda la documentación y procedemos a valorar la cuenta de resultados.

Vamos a suponer que disponemos de 20.000 dólares. Introducimos las variables correspondientes en nuestro programa informático y extraemos el siguiente gráfico:



Con los resultados facilitados por el gestor que queremos contratar, se han realizado 10.000 simulaciones. Además, los resultados se han proyectado cuatro años. Esto es, 10.000 escenarios diferentes para esos resultados durante cuatro años. En la gran mayoría de escenarios se genera una rentabilidad positiva, pero existe una pequeña probabilidad de perder dinero. La simulación de Montecarlo nos facilita una infinidad de combinaciones para evaluar escenarios de los que a simple vista no somos conscientes [1].

Ventajas y desventajas

La simulación Monte Carlo proporciona una serie de ventajas sobre el análisis determinista o "estimación de un solo punto":

- Resultados probabilísticos. Los resultados muestran no sólo lo que puede suceder, sino lo probable que es un resultado.
- Resultados gráficos. Gracias a los datos que genera una simulación Monte Carlo, es fácil crear gráficos de diferentes resultados y las posibilidades de que sucedan. Esto es importante para comunicar los resultados a otras personas interesadas.
- Análisis de sensibilidad. Con sólo unos pocos resultados, en los análisis deterministas es más difícil ver las variables que más afectan el resultado. En la simulación Monte Carlo, resulta más fácil ver qué variables introducidas tienen mayor influencia sobre los resultados finales.
- Análisis de escenario. En los modelos deterministas resulta muy difícil modelar diferentes combinaciones de valores de diferentes valores de entrada, con el fin de ver los efectos de situaciones verdaderamente diferentes. Usando la simulación Monte Carlo, los analistas pueden ver exactamente los valores que tienen cada variable cuando se producen ciertos resultados. Esto resulta muy valioso para profundizar en los análisis.
- Correlación de variables de entrada. En la simulación Monte Carlo es posible modelar relaciones interdependientes entre diferentes variables de entrada. Esto es importante para averiguar con precisión la razón real por la que, cuando algunos factores suben, otros suben o bajan paralelamente [2].

Entre sus desventajas se puede establecer:

- Una buena simulación puede resultar muy complicada, gran numero 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 [3].

7 DISCUSIÓN

Mediante el estudio del lenguaje R podemos aprender métodos para realizar análisis de datos, pero sobre todo representar gráficamente esos análisis de datos estadísticos como media, varianza y desviación estándar resultan ser muy sencillos de efectuar en un conjunto de variables aleatorias, especialmente si se cuenta con la herramienta necesaria para realizar y reducir el trabajo en dicha actividad.
En este quinto laboratorio se realiza una investigación complementaria sobre la simulación de Montecarlo ofrece a la persona responsable de tomar las decisiones una serie de posibles resultados, así como la probabilidad de que se produzcan según las medidas tomadas. Muestra las posibilidades extremas los resultados de tomar la medida más arriesgada y la más conservadora, así como todas las posibles consecuencias de las decisiones intermedias.

8 CONCLUSIONES

- Al haber concluido esta práctica se tiene conocimiento de las herramientas básicas para utilizar R como un lenguaje de programación. Desde los conceptos más básicos, hasta la definición de funciones, además R nos permite el cálculo de predicciones basados en estadísticas, no solo mediante código sino gráficamente, lo que en otros lenguajes de programación resultaría un código muy extenso y complejo.
- La generación de números pseudoaleatorios en R es una de las mejores disponibles en paquetes estadísticos.
- Se llega a concluir que R un lenguaje potente con un gran objetivo orientado a la estadística es por eso que si nos profundizamos en el podremos conocerlo y analizarlo más, para así poder aplicarlo en nuestra carrera.

9 RECOMENDACIONES

- Para programar en R, necesitamos conocer el uso de sus funciones, el tipo objeto al que puede aplicárseles, sus argumentos, así como los valores que requieren ser definidos.
- Utilizar los métodos proporcionados por el lenguaje R para el cálculo de valores verdaderos
- Revisar la interfaz gráfica del programa para así poder conocerlo y poder compartir opiniones y experiencias en clases.

BIBLIOGRAFÍA:

[1] J. F. Lopez, «Economipedia "Simulación de Montecarlo",» [En línea]. Available: https://economipedia.com/definiciones/simulacion-de-montecarlo.html. [Último acceso: 17 Diciembre 2019].

[2] Palisade, «Palisade "Simulación Monte Carlo",» [En línea]. Available: https://www.palisade-lta.com/risk/simulacion_monte_carlo.asp. [Último acceso: 17 Diciembre 2019].

[3] A. Meneses, «MÉTODO MONTECARLO, ORIGEN, VENTAJAS Y DESVENTAJAS,» 27 agosto 2012. [En línea]. Available: http://unimeta-simulacion-alejandra-meneses.blogspot.com/2012/08/metodo-origen-el-de-montecarlo-un-no.html. [Último acceso: 17 Diciembre 2019].