Facultad de Ingeniería, Universidad de la República

Montevideo, Uruguay

INFORME LABORATORIO 1

Métodos de Monte Carlo

Autor

Francisco Casarotti - 4.595.179-7

Supervisor

Ing. Hector Cancela

16/03/2021

ÍNDICE

Introducción

Se puede decir que existen dos tipos de matemáticos, los teóricos y los experimentales. La diferencia entre ambos, es que unos deducen sus conclusiones de postulados y otros de observaciones, respectivamente.

Los métodos de Monte Carlo comprenden la rama de las matemáticas experimentales que se ocupa de los experimentos con números aleatorios. Este método es utilizado para la resolución de dos tipos de problemas: probabilísticos o determinísticos. En el primer caso, simplemente se observan números aleatorios (que simulan el procesos asociado al problema original) y de ahí se infiere una solución al problema. El segundo caso, es un poco más complejo. La idea es "explotar" las fortalezas de la de las matemáticas teóricas y al mismo tiempo evitar sus "debilidades" mediante la utilización de experimentos cuando sea necesario.

Se proporcionan soluciones aproximadas a problemas matemáticos mediante la realización de experimentos de muestreo estadístico en una computadora. Es posible aplicar tanto a problemas sin ningún contenido probabilístico en absoluto, así como a aquellos con una estructura probabilística inherente. Esto de por si no le brinda una ventaja por sobre métodos, sin embargo, de todos los métodos que se basan en n-puntos de m-dimensiones, este tiene un error absoluto de estimación que disminuye $n^{-½}$,mientras que el resto tiene errores que decrecen como $n^{-1/m}$ (en el mejor de los casos).

El nombre surgió en los alrededores de 1944, pero hay indicios de que su aplicación data de tiempo antes. Por ejemplo, en la segunda mitad del siglo XIX, un conjunto de personas llevaron a cabo un experimento en el cual lanzaban una aguja sobre un tablero con líneas rectas paralelas, luego infirieron de ello el valor de π a partir de las intersecciones entre las agujas y las líneas. Pero el verdadero uso de Monte Carlo llegó con la bomba atómica, este trabajo involucró una simulación directa de los problemas probabilísticos relacionados con la difusión aleatoria de neutrones en material fisionable.

De forma más precisa, el método de Monte Carlo incorpora tres desarrollos históricos distintos, pero relacionados.

1) Primero, los juegos de azar motivaron a los matemáticos de los siglos XVII y XVIII. Posteriormente los estadísticos del siglo XIX y principios del XX reconocieron que, uno podía sacar números al azar, transformarlos de acuerdo a ciertas reglas y derivar un solución aproximada a una integral en un problema que intrínsecamente no contenía ningún contenido probabilístico.
2) A fines del siglo XIX, se desarrolló una segunda línea de investigación cuando Lord Rayleigh (1899) mostró que un camino aleatorio unidimensional sin barreras absorbentes podría brindar una solución aproximada a una ecuación diferencial parabólica.

   En el curso de la demostración de que, en condiciones apropiadas, una ecuación de diferencia finita parcial podría producir una solución aproximada al problema de valores en la frontera de Dirichlet de las ecuaciones diferenciales parciales, Courant et al. (1928) mostró que la forma recursiva de la solución de un camino aleatorio bidimensional en una cuadrícula dentro de una región cerrada (cuyos puntos límite eran estados absorbentes), producía la misma "difference equation" (ecuación de diferencias).

   Poco después, Kolmogorov (1931) mostró la relación entre los procesos estocásticos de Markov y ciertas ecuaciones integro-diferenciales. Petrowsky (1933) generalizó considerablemente el resultado de Courant et al. mostrando la conexión asintótica entre un camino aleatorio (cuya secuencia de ubicaciones formaba una cadena de Markov) y la solución a una ecuación

diferencial parcial elíptica. Llamó a esta formulación el problema generalizado de Dirichlet.

Se empezó a observar que la solución a problemas encontrados en procesos estocásticos generalmente se correspondían con soluciones que abordan el método de ecuaciones de diferencias para la resolución de ecuaciones diferenciales, y así proveer una solución aproximada a estos problemas.

3) Durante el desarrollo de la energía atómica en la era posterior a la Segunda Guerra Mundial, se desarrolló una tercera línea de investigación. Los científicos necesitaban resolver problemas de difusión o transporte de neutrones a través de un medio isotrópico. Estos problemas multidimensionales resultaron demasiado formidables para el enfoque de las ecuaciones de diferencias.

   John von Neumann y Stanislaw Ulam sugirieron la utilización de experimentos de muestreo, utilizando modelos de caminos aleatorios y ejecutados en la computadora digital recién desarrollada, afirmando que estos podrían proporcionar aproximaciones fácilmente utilizables a la soluciones deseadas. Esta propuesta invirtió la dirección del razonamiento. En lugar de utilizar las ecuaciones de diferencia para proporcionar soluciones a los problemas, ahora se realizan experimentos de muestreo para proporcionar soluciones a las ecuaciones integro-diferenciales, que no necesariamente tenían una base probabilística en sí mismas.

   Más tarde, el concepto de emplear experimentos de muestreo en una computadora prevaleció en muchas disciplinas científicas, sobre todo en química, ingeniería, investigación de operaciones, física y estadística. En la investigación de operaciones, los estudios de todos los modelos de colas, excepto los más elementales, se basan en la generación de recorridos aleatorios en las computadoras para proporcionar soluciones aproximadas. En física, el método de Monte Carlo ha llegado a ser reconocido como el único enfoque capaz de proporcionar información útil para muchos problemas de transporte de neutrones y estructuras reticulares.

Bases del Método de Monte Carlo

Es claro que este tipo de método no es capaz de proveer una solución exacta a un problema, sino una solución aproximada al mismo. La justificación para emplear el método Monte Carlo proviene de dos teoremas centrales de la probabilidad y estadística: la "Ley Débil de los Grandes Números" y el "Teorema Central del Límite".

Sea $X_1, X_2, \ldots X_n$ "n" variables aleatorias independientes e idénticamente distribuidas. Sea $S_n = X_1 + X_2 + \ldots + X_n$. Si la esperanza existe, denotada como $\boldsymbol{\mu} = E(X_i)$, entonces la "Ley Débil de los Grandes Números" indica que, para cada $\boldsymbol{\varepsilon} > 0$

$$\lim_{n \to \infty} Pr\left(\left|\frac{Sn}{n} - \mu\right| > \varepsilon\right) = 0$$

Esta Ley lleva a creer que a medida que aumenta el tamaño de la muestra, el error en la estimación de μ mediante S_n/n se vuelve cada vez más pequeño.

Si, adicionalmente, existe la varianza $\sigma^2 = E(X_i - \mu)^2$, entonces el "Teorema Central del Límite" establece que, para un valor "a" fijo

$$\lim_{n \to \infty} Pr(\frac{Sn - n\mu}{\sigma\sqrt{n}} < a) = (2\pi)^{-1/2} \int_{-\infty}^{a} e^{-z^2/2}$$

En principio, el Teorema del Límite Central proporciona una forma de evaluar la extensión del ese error estadístico (estimar μ mediante S_n/n) para n grande.

En la práctica, el método Monte Carlo se basa en técnicas que contradicen ambas leyes estadísticas. Dado que todas las muestras con valores numéricos generadas en un experimento de Monte Carlo son producidas por un generador de números pseudoaleatorios, y dado que las secuencias de estos números se repiten después de un número finito de pasos P, el muestreo sin límite en un experimento de Monte Carlo no hace que el error estadístico desaparezca. Von Neumann advertía sobre los métodos usados para los pseudoaleatorios: "Cualquiera que considere métodos aritméticos para producir dígitos aleatorios comete, por supuesto, un pecado mortal". Algunos de estos generadores son: Método de los cuadrados medios, generador de Lehmer y generador de Wichmann–Hill. Cada cual tiene su propio algoritmo determinista mediante el cual calcula número pseudo-aleatorios.

El Método de los medios cuadrados fue planteado por Metropolis y Von Neumann, en el cual cada número es generado elevando al cuadrado su predecesor y "tomando" los dígitos medios. Por así decirlo, x_{i+1} consiste en los medios dígitos de x_i^2. Más adelante se demostró que este método es insatisfactorio. En 1951, Lehmer sugirió que se podía llegar a una secuencia pseudo aleatoria de números mediante una recurrencia: $x_i \cong ax_{i-1}(\text{ mod } m)$.

Pseudocodigo de Monte Carlo

Dada una variable aleatoria X con distribución Fx, esperanza μ=E(X) y varianza σ^2=E[(X-μ)^2], se puede utilizar Monte Carlo para estimar estos valores.

Para la estimación de μ se utiliza la muestra promedio (Sample Mean).
Sean $X^{(1)}, X^{(2)}, \ldots, X^{(n)}$ numeros aleatorios independientes (de la variable aleatoria X), el muestreo promedio se calcula como:

$$var(X) = S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x^{(i)} - \bar{x})^2$$

Desarrollando el término se llega a:

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x^{(i)^2} - 2x^{(i)}\bar{x} + \bar{x}^2)$$

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x^{(i)^2} - 2\bar{x} \sum_{i=1}^{n} x^{(i)} + \sum_{i=1}^{n} \bar{x}^2)$$

Cálculo de la varianza del estimador insesgado:

$$var(\overline{X}) = \frac{var(X)}{n} = \frac{1}{n}\frac{1}{n-1}\left(\sum_{i=1}^{n} x^{(i)^2} - 2\overline{x}\sum_{i=1}^{n} x^{(i)} + \sum_{i=1}^{n} \overline{x}^2\right)$$

Tomando las siguientes consideraciones:

$$\sum_{i=1}^{n} x^{(i)} = \overline{x}n$$

$$\sum_{i=1}^{n} \overline{x}^2 = \overline{x}^2 n$$

y sustituyendo se llega a lo siguiente:

$$var(\overline{X}) = \frac{1}{n}\frac{1}{n-1}\left(\sum_{i=1}^{n} x^{(i)^2} - 2\overline{x}^2 n + \overline{x}^2 n\right)$$

$$var(\overline{X}) = \frac{1}{n}\frac{1}{n-1}\left(\sum_{i=1}^{n} x^{(i)^2} - \overline{x}^2 n\right)$$

$$var(\overline{X}) = \frac{\sum_{i=1}^{n} x^{(i)^2}}{n(n-1)} - \frac{\overline{x}^2}{n-1}$$

A partir de estos cálculos, se puede desarrollar un pseudo código más eficiente para el cálculo de la esperanza, varianza y desviación media (se calcula con la varianza).

Procedimiento EstimaciónMonteCarlo (integer n, real $\widehat{X}$, real $\widehat{V}$)
Parámetro de entrada: n, *tamaño de la muestra*
Parámetros de salida: $\widehat{X}$, *estimador* de ϕ; $\widehat{V}$, *estimador* de Var $\left(\widehat{X}\right)$

1. $\widehat{X} = 0.$ /* Inicialización */
2. $\widehat{V} = 0.$
3. For $i = 1, \ldots, n$ do
   3.1 Sortear un valor de la variable $X^{(i)}$ con distribución F_X
   3.2 $\widehat{X} = \widehat{X} + X^{(i)}$ /* Acumular*/
   3.3 $\widehat{V} = \widehat{V} + (X^{(i)})^2$ /* Acumular*/
4. $\widehat{X} = \widehat{X}/n$
5. $\widehat{V} = \widehat{V}/(n * (n-1)) - \widehat{X}^2/(n-1)$

Problema - “Queremos construir una casa”

El problema a abordar es estimar el tiempo empleado para la construcción de una cierta casa. Se conocen los rangos de tiempos (regidos por variables aleatorias uniformes) de cada una de las tareas requeridas para dicha construcción, y las ataduras que existen entre cada una de esas tareas (ciertas tareas no pueden empezar antes que terminen otras).

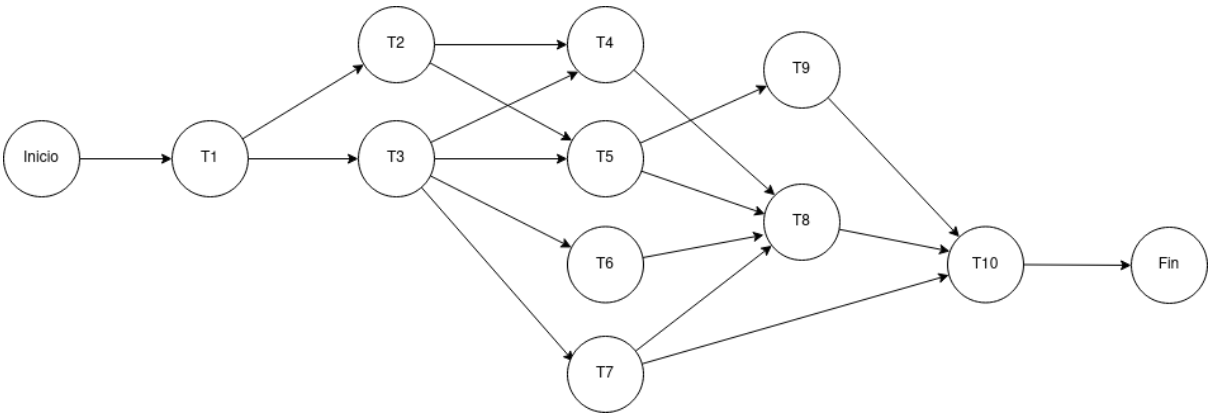
Las tareas y sus respectivos tiempos son los siguientes:

Tarea	Descripción	Tiempo aleatorio uniforme entre
T1	Cimientos	40 y 56 hs
T2	Contrapiso	24 y 32 hs
T3	Caredes	20 y 40 hs
T4	Techo	16 y 48 hs
T5	Instalación sanitaria	10 y 30 hs
T6	Instalación eléctrica	15 y 30 hs
T7	Cerramientos	20 y 25 hs
T8	Pintura	30 y 50 hs
T9	Revestimientos sanitarios	40 y 60 hs
T10	Limpieza final	8 y 16 hs

Las dependencias (que implican que una tarea no puede comenzar hasta haberse terminado otra previa) son las siguientes:

Tarea	Depende de
T2, T3	T1
T4	T2, T3
T5	T2, T3
T6	T3
T7	T3
T8	T4, T5, T6, T7
T9	T5
T10	T7, T8, T9

Se puede visualizar las dependencias de las tareas utilizando un grafo:



Las ponderaciones de las aristas son las horas que tarda cada tarea, por ejemplo, a las aristas salientes de T1 les corresponde el tiempo que tarda T1. A priori no está determinado sino que son valores aleatorios (en ciertos rangos), razón por la cual no se encuentra ilustrado en el grafo dirigido.

Solución al problema

Para la resolución de este problema se puede utilizar Monte Carlo. Una primera aproximación a la resolución podría ser la de sortear los valores de las tareas en cada replicación, y calcular el camino crítico de esta, pero tiene un costo computacional muy elevado. Estudiando este caso en particular, y viendo que es un número relativamente chico de tareas, se podría calcular los posibles caminos críticos (utilizando Monte Carlo), para luego estimar con un número elevado de replicaciones la cantidad de horas de la construcción de la casa.

Posibles caminos críticos

Para el cálculo de los caminos críticos se utilizó la librería de “Python” llamada “criticalpath”.

En una primera instancia se setean las tareas, duraciones (por defecto en uno) y las dependencias existentes entre ellas.

```
tareas = [("T1", {"duracion": 1}),
          ("T2", {"duracion": 1}),
          ("T3", {"duracion": 1}),
          ("T4", {"duracion": 1}),
          ("T5", {"duracion": 1}),
          ("T8", {"duracion": 1}),
          ("T9", {"duracion": 1}),
         ("T10", {"duracion": 1})]
```

```
dependencias = [("T2", "T1"),
                ("T3", "T1"),
                ("T4", "T2"),
                ("T4", "T3"),
                ("T5", "T2"),
                ("T5", "T3"),
                ("T8", "T4"),
                ("T8", "T5"),
                ("T9", "T5"),
               ("T10", "T8"),
          ("T10", "T9")]
```

Luego se definen los tiempos que regirán las variables aleatorias.

```
tiempos = [[40,56],
           [24,32],
           [20,40],
           [16,48],
           [10,30],
           [30,50],
           [40,60],
           [8,16],
          ]
```

En cada replicación, se sortean los valores de T1, … , T10 y son pasados como el parámetro "t" de la siguiente función. Esta retorna en forma de arreglo el camino crítico y su duración.

```
def camino_critico(tareas,dependencias,t):

    p = Node('proyecto')

    for i in range(0,8):
        tareas[i][1]['duracion'] = t[i]

    for i in tareas:
        p.add(Node(i[0], duration=i[1]["duracion"]))

    for j in dependencias:
        p.link(j[0],j[1])

    p.update_all()

    return [p.duration, p.get_critical_path()]
```

Ahora se definen dos funciones más, "monte_carlo_count" y "ejecuciones_de_tareas" donde la segunda llama a la primera.

La función "monte_carlo_count" se encarga de calcular los caminos críticos que son generados en n replicaciones, y la cantidad de veces que son "usados" en esas n replicaciones.

Sus parámetros son "tareas","dependencias","t" y "n". Las dos primeras son para pasarlas a la funcion "camino_critico", mientras que "t" es el parámetro de los tiempos uniformes (se le pasa el arreglo de "tiempos") y "n" es la cantidad de replicaciones que se quieren ejecutar.

```
def monte_carlo_count(tareas,dependencias,t,n):
    duraciones = []
    for j in range(0,10):
        duraciones = np.append(duraciones,0)

    posibles_caminos = {}

    np.random.seed(12345)
    for i in range(0,n+1):
        caminos = []
        for j in range(0,10):
            valor = np.random.uniform(t[j][0],t[j][1])
            caminos = np.append(caminos,valor)

        valor = camino_critico(tareas,dependencias,caminos)

        elem = ''
        for nomb in valor[1]:
            elem = elem + str(nomb) + '<-'
        elem = elem[:-2]

        if elem in posibles_caminos:
            posibles_caminos[elem] += 1
        else:
            posibles_caminos[elem] = 1
        for nomb in valor[1]:
            indice = int(re.sub(r'^T','',str(nomb))) - 1
            duraciones[indice] += 1

    return [duraciones, posibles_caminos]
```

"ejecuciones_de_tareas" tiene como único parámetro "n", en donde se le pasa la cantidad de replicaciones que se desean realizar.

```
def ejecuciones_de_tareas(n):
    valores = monte_carlo_count(tareas,dependencias,tiempos,n)
    nombre_tareas = []
    for j in range(1,11):
            nombre_tareas = np.append(nombre_tareas,'T'+str(j))
    df_tareas = pd.DataFrame(nombre_tareas, columns=['Tarea'])
    df_duraciones = pd.DataFrame(valores[0], columns=['#Pertenencias al camino crítico'])
    porcentajes = (valores[0]/n)*100
    df_porcentajes = pd.DataFrame(porcentajes, columns=['% de pertenencias'])

    df = pd.concat([df_tareas, df_duraciones,df_porcentajes], axis = 1)

    df_caminos = pd.DataFrame(list(valores[1].items()),columns=['Caminos', '#de ejecuciones'])

    porcentajes_caminos = (df_caminos.iloc[:,1].values/n)*100
    df_porcentajes_caminos = pd.DataFrame(porcentajes_caminos ,columns=['% de ejecuciones'])

    df_caminos = pd.concat([df_caminos,df_porcentajes_caminos], axis = 1)
    return [df,df_caminos]
```

Con 100.000 replicaciones se obtienen los siguientes resultados.

Tarea	#Pertenencias al camino crítico	% de pertenencias
T1	100000.0	100.000
T10	100000.0	100.000
T3	61395.0	61.395
T8	58983.0	58.983
T4	51368.0	51.368
T5	42378.0	42.378
T9	41017.0	41.017
T2	38605.0	38.605
T6	4415.0	4.415
T7	1839.0	1.839

Caminos	#de ejecuciones	% de ejecuciones
T10<-T8<-T4<-T3<-T1	30189	30.189
T10<-T9<-T5<-T3<-T1	24281	24.281
T10<-T8<-T4<-T2<-T1	21179	21.179
T10<-T9<-T5<-T2<-T1	16736	16.736
T10<-T8<-T6<-T3<-T1	4415	4.415
T10<-T8<-T7<-T3<-T1	1839	1.839
T10<-T8<-T5<-T2<-T1	690	0.690
T10<-T8<-T5<-T3<-T1	671	0.671

Resolución

Al ser un número reducido de posibles caminos críticos, una forma de aprovechar el poder computacional, es calcular en cada replicación cual de estos caminos posee el mayor valor. Luego se invoca la función y se calcula el tiempo de ejecución de la misma.

```
start_time = time.time()
e,v,dm = monte_carlo(tiempos,n)
tiempos_exec = time.time() - start_time
```

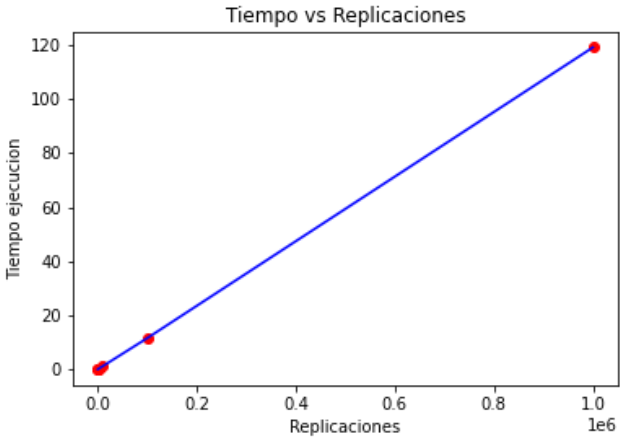
Este es un ejemplo para 10^5 replicaciones

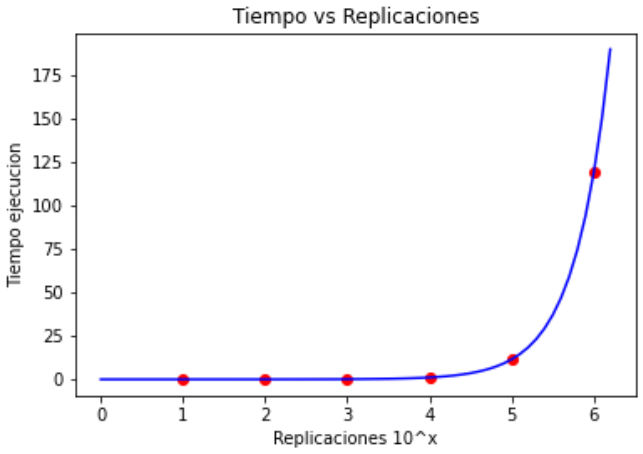
```
Esperanza -> 168.554 horas,
Varianza -> 0.0010624 horas
Desviacion Media -> 0.0325938 horas
El tiempo de ejecucion de la funcion es 11.26 segundos
```

Se realizó este mismo proceso para 10, 10^2, … , 10^6 y se obtuvieron los siguientes resultados.

```
10^1 replicaciones -> 0.007 seg
10^2 replicaciones -> 0.026 seg
10^3 replicaciones -> 0.117 seg
10^4 replicaciones -> 1.110 seg
10^5 replicaciones -> 11.522 seg
10^6 replicaciones -> 119.208 seg
```

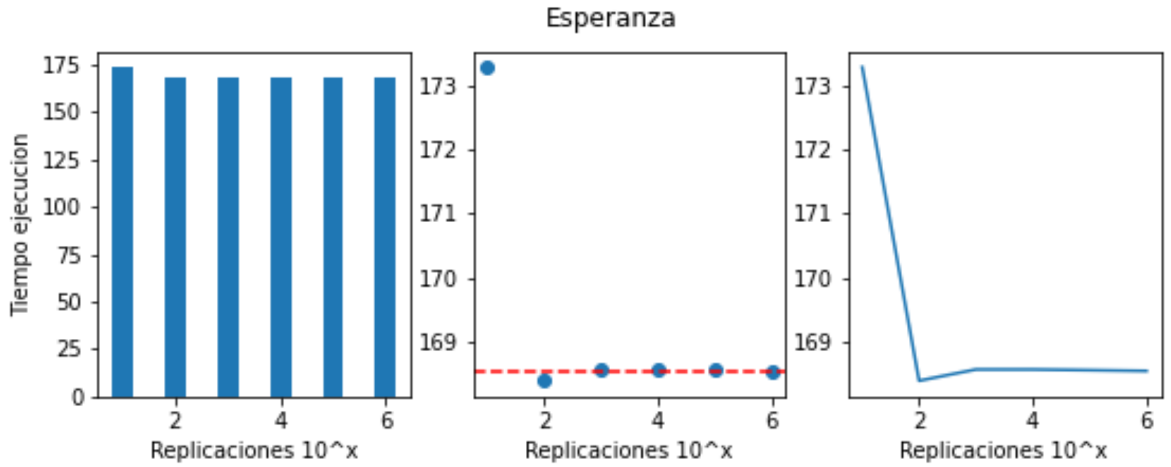
Graficando se obtiene.



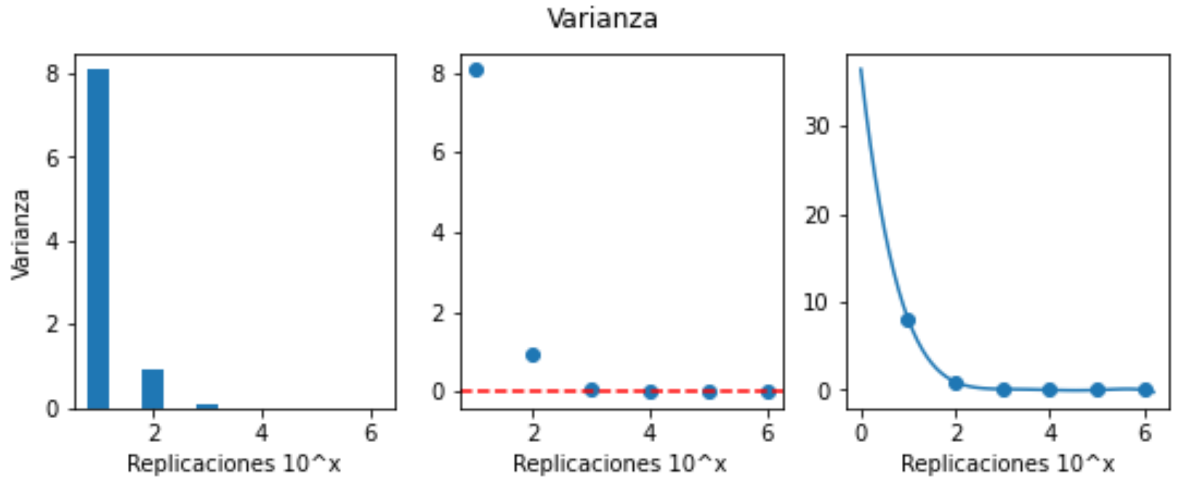


El tiempo de ejecución del método en relación a la cantidad de replicaciones, es claramente lineal.

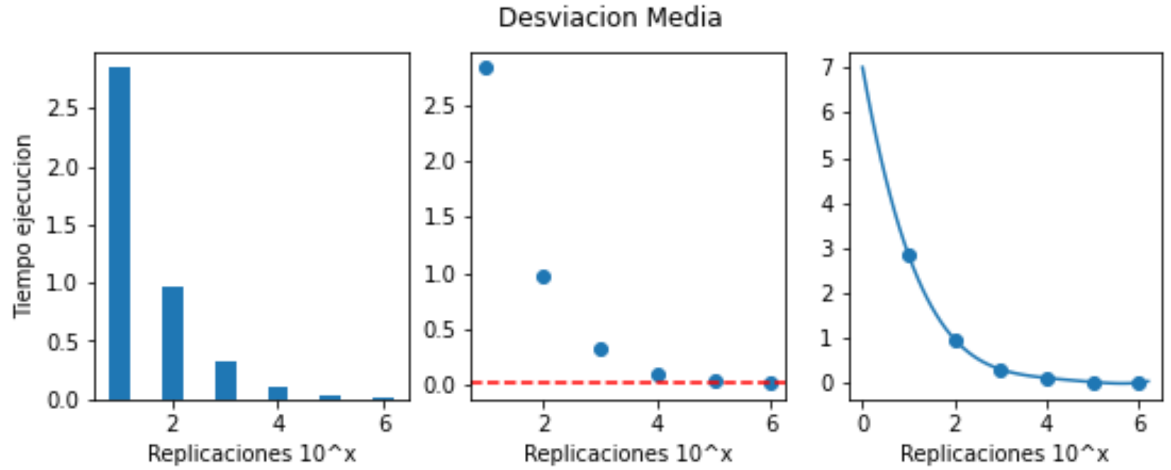
En cuanto a la Esperanza, se obtienen las siguientes gráficas.



En cuanto a la Varianza.



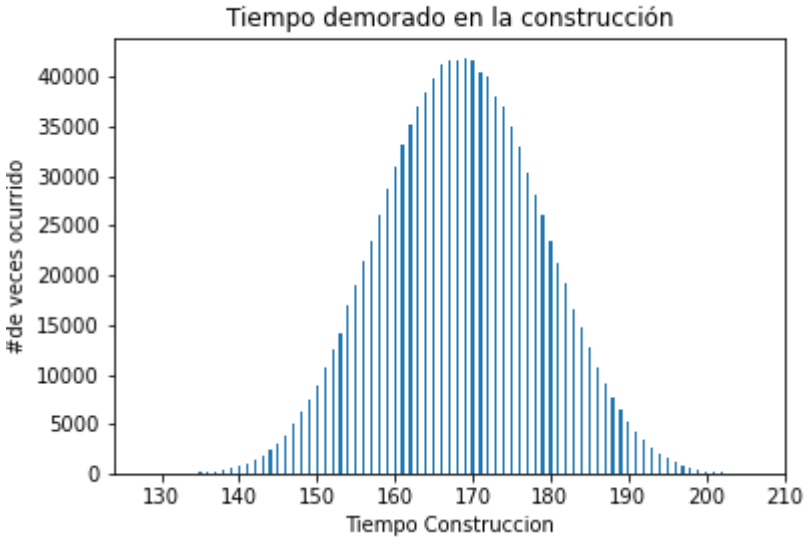
En cuanto a la Desviación Media



En los tres casos se puede apreciar una estabilización asintótica hacia un valor dado. Para el caso de la Esperanza, esta se aproxima al valor cercano a 168.54230157, mientras que la Varianza es de 0.00010616 y la Desviación Media tiende a 0.01030370.

Conclusiones

En la siguiente gráfica se puede visualizar la distribución de los tiempos de la construcción en función de la cantidad de veces que se incurrió en cada tiempo, esto en 10^6 replicaciones.



Dadas las condiciones iniciales para la construcción y realizando simulaciones de hasta 10^6 replicaciones, se puede modelar esta situación como una distribución normal, con una Esperanza de 168.54230157 horas y Desviación Media de 0.01030370 horas.