Introducción

El método Monte Carlo es idóneo para situaciones en las cuales algún valor o alguna distribución no se conoce y resulta complicado de determinar de manera analítica.

Meta

Examinar el efecto del tamaño de muestra en la precisión del estimado y medir el tiempo de ejecución para cada tamaño de muestra para resolver la integral

$$\int f(x) = \int_3^7 \frac{1}{e^x + e^{-x}}$$

Código

```
> library(ggplot2)
> inicio <- -6
> final <- -inicio
> paso <- 0.25
> ciclos<-50
> varcuan<-seq(10,100,10)
> x <- seq(inicio, final, paso)
> f <- function(x) { return(1 / (exp(x) + exp(-x))) }
>
> suppressMessages(library(distr))
> g <- function(x) { return((2 / pi) * f(x)) }
> generador  <- r(AbscontDistribution(d = g)) # creamos un generador
>
> wa<-0.0488341111
> desde <- 3
> hasta <- 7
> pedazo <- 50000
> res<-data.frame()
>
> parte <- function() {
+   valores <- generador(pedazo)
+   return(sum(valores >= desde & valores <= hasta))
+ }
> suppressMessages(library(doParallel))
> registerDoParallel(makeCluster(detectCores() - 1))
>
> for (cuantos in varcuan){
+   n<-cuantos*pedazo
+
+   for (replica in 1:ciclos){
+
+ t<-system.time({
+
+ montecarlo <- foreach(i = 1:cuantos, .combine=c) %dopar% parte()
+ integral <- sum(montecarlo) / n
```

```
+ int<-(pi / 2) * integral
+ })
+ error<-abs(int-wa)
+ e<-cbind(int,error, cuantos,t[3])
+ res<-rbind(res,e)
+
+   }
+ }
>
> stopImplicitCluster()
>
>
> colnames(res)<-c("valor","error","cuantos","tiempo")
>
> png("Variacion_error.png")
> res$cuantos<-as.factor(res$cuantos)
> ggplot(data=res,aes(x=res$cuantos,y=res$error))+
+   geom_boxplot(fill="cadetblue2")+
+   scale_y_continuous(name="Error",labels=scales::comma) +
+   scale_x_discrete(name="Muestras")
>
> png("Variacion_tiempo.png")
> ggplot(data=res,aes(x=res$cuantos,y=res$tiempo))+
+   geom_boxplot(fill="cadetblue2")+
+   scale_y_continuous(name="Tiempo",labels=scales::comma) +
+   scale_x_discrete(name="Muestras")
> graphics.off()
> rm(list=ls())
> gc()
```

Desarrollo del Código

El código R presentado en la sección “Código” es resultado de modificaciones (en color azul) al código R ejemplificado en clase [1] con los siguientes objetivos:

- Llamar a la librería *ggplot2* con el objeto de crear gráficos atractivos.
- Modificar de manera rápida el número de repeticiones “ciclos” para repetir un experimento bajo las mismas condiciones y además de modificar la secuencia de la variable “cuantos” que modifica el tamaño de la muestra a analizar.
- Establecer el valor más cercano al valor real de la solución de la integral. Este valor es obtenido en línea [2].
- Declarar un *data.frame* “res” para guardar de manera ordenada los resultados de la simulación.
- Generar un “loop” para diferentes números de tamaño de muestra “varcuan” y generar una serie de repeticiones acorde a la variable “ciclos”.
- Crear una variable “t” para guardar el tiempo de ejecución de la tarea asignada.

- Realizar la operación aritmética de resta entre el valor calculado y la aproximación más cercana.
- Generar columnas con los valores de distintas variables.
- Asignar un nombre a las columnas del *data.frame* “res”.
- Crear gráficos para mostrar los resultados obtenidos.

Resultados

En la figura 1 muestra el valor absoluto de la diferencia aritmética entre el valor obtenido durante la simulación y el valor obtenido del portal en línea [2]. En esta representación se puede observar que el tamaño de las cajas disminuye conforme se aumenta el tamaño de la muestra, es decir, los valores de ambas comparaciones son más similares por lo cual tienden a estar dentro de un rango más pequeño. Se dice “pequeño” en comparación de otros tamaños de muestra, sin embargo, si se observan los valores de esta diferencia es posible ver que éstas se encuentran en el orden de las diezmilésimas, lo que recae en una gran exactitud del método, es decir se aproxima al valor real.

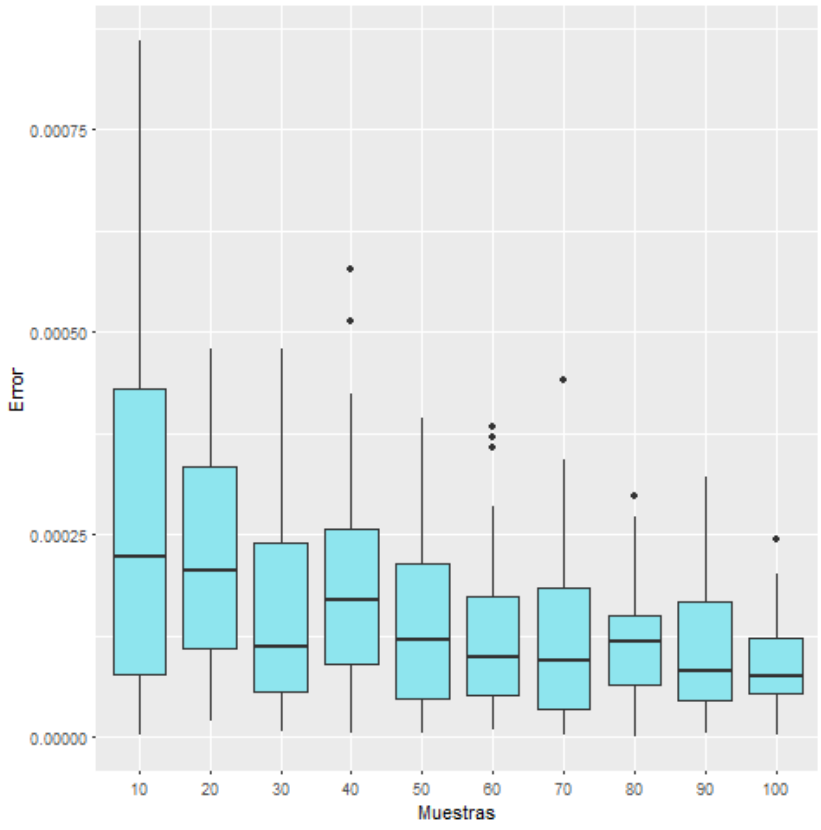

Figura 1 Representación gráfica de la diferencia aritmética entre el valor de la integral calculado experimentalmente y el de la integral teórica

Si bien se desea disminuir aún más esta diferencia, se debe tener en consideración el tiempo necesario para conseguir este objetivo. En la figura 2 se puede observar que aumentar el tamaño de la muestra aumenta el tiempo de ejecución de la tarea asignada, es decir existe una relación directamente proporcional entre ambos. En la figura 2, se observa que la relación entre el tiempo de ejecución y el tamaño de muestra se parece mucho a una línea recta, por lo cual podría extrapolarse dicha recta, para conocer el tiempo que tardaría realizar el mismo código para un tamaño de muestra más grande. Y así poder decidir si en verdad será útil sacrificar un poco más en el tiempo de ejecución para poder ganar una mayor exactitud en la resolución de la integral. Se observa también, que el largo de la "caja" de cada tamaño de muestra es relativamente pequeño, es decir, el procedimiento está definido de tal manera que no existe un gran margen para variar notablemente el tiempo de ejecución.

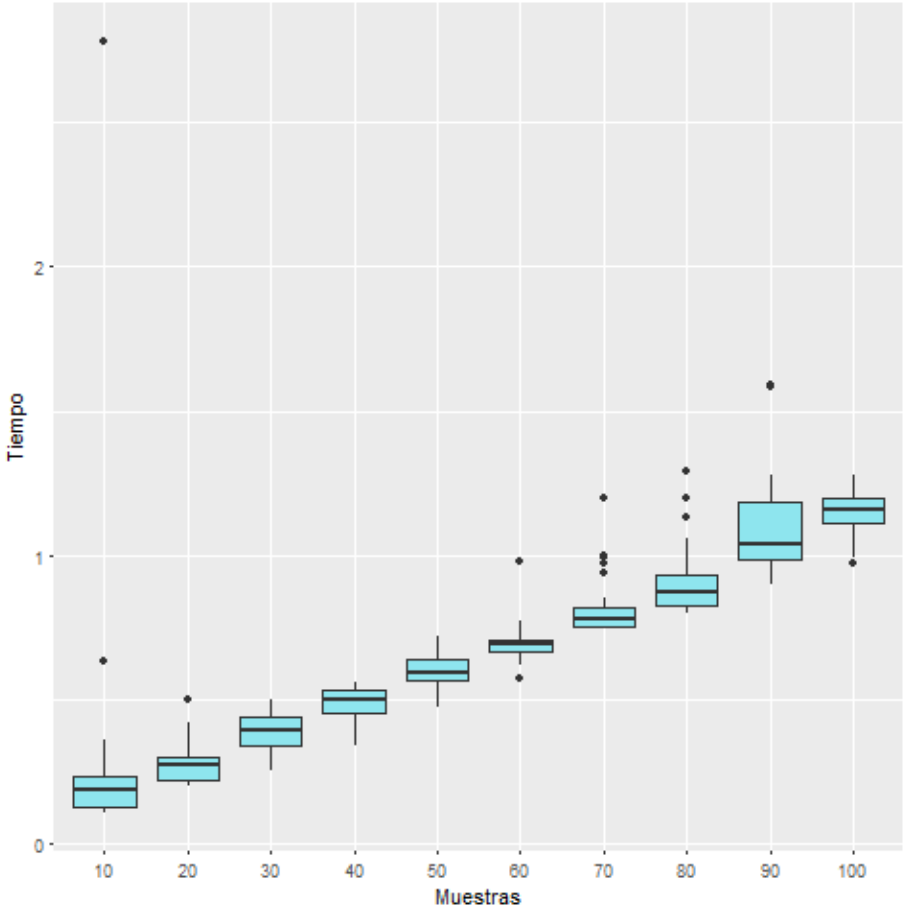

Figura 2 Representación de la variación en el tiempo de ejecución en función al tamaño de muestra

Introducción Reto 1

Dentro del conjunto de números reales, existen aquellos conocidos como los números irracionales, los cuales no pueden ser representados por una fracción, a causa de que sus números decimales no siguen algún patrón. Dentro de estos números se encuentran dos constantes muy conocidas: el número de Euler y π. En nuestra trayectoria académica se nos ha dado a conocer una aproximación aceptable de ambos números, pero debemos tener en claro que, si bien se acercan mucho al valor real, estas aproximaciones no son correctas.

En esta práctica se realizará una aproximación de la constante π utilizando las fórmulas de área de un círculo de radio *r* (Eq. 1) y de un cuadrado de lado *2r* (Eq. 2) .

$$A_{círculo} = \qquad \text{(Eq. 1)}$$

$$A_{cuadrado} = 4r^2 \qquad \text{(Eq. 2)}$$

La figura 3 representa de manera gráfica la razón entre ambas áreas. La cual matemáticamente se representa

$$Razón = \frac{\pi r^2}{4r^2} = \frac{\pi}{4} \qquad \text{(Eq.3)}$$

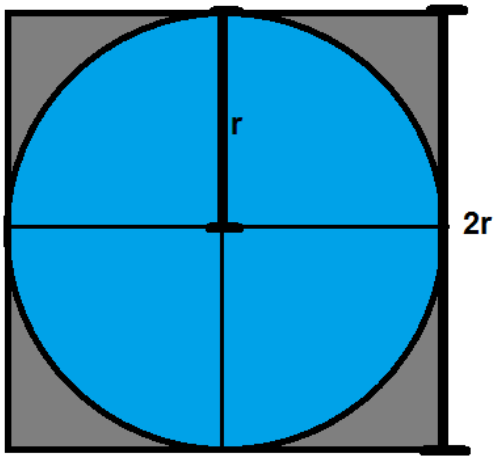

Figura 3 Representación gráfica del área de un círculo de radio r, dentro de un cuadrado de lado 2r.

Metas

En el Reto 1 se cuenta con 3 metas:

- Implementar la estimación del valor de π de Kurt [3]con paralelismo.
- Examinar el efecto del tamaño de muestra en la precisión del estimado.
- Medir el tiempo de ejecución para cada tamaño de muestra.

Código

```
> library(ggplot2)
> library(parallel)
>
> varruns<-seq (100000,700000,100000)
> ciclos<-40
>
> dat<-data.frame()
> res<-data.frame()
>
> rpi<-3.1415926535
> digpi<-as.matrix(substring(rpi,seq(nchar(rpi)),seq(nchar(rpi))))
>
> cluster <- makeCluster(detectCores() - 1)
>
> for (runs in varruns){
+
+  for (replica in 1:ciclos){
+
+ t<-system.time({
+ xs <- runif(runs,min=-0.5,max=0.5)
+ ys <- runif(runs,min=-0.5,max=0.5)
+ in.circle <- xs^2 + ys^2 <= 0.5^2
+ mc.pi <- (sum(in.circle)/runs)*4
+   }
+ )
+
+ digmc<-matrix(substring(mc.pi,seq(nchar(mc.pi)),seq(nchar(mc.pi))))
+ for (new in 1:abs((length(digpi)-length(digmc)))){
+   digmc<-rbind(digmc,0)
+ }
+
+ d<-numeric()
+
+ for (p in 3:12){
+   if (digpi[p]== digmc[p]){d<-c(d,TRUE)}
+   else
+     break;
+ }
+
+ igu<-sum(d)
+ dif<-abs(mc.pi-rpi)
```

```
+ dat<-cbind(mc.pi,(runs/100000),igu,dif,t[3])
+ res<-rbind(res,dat)
+
+   }
+ }
>
> stopCluster(cluster)
>
> colnames(res)<-c("Valor","Muestra","Presicion","Diferencia","Tiempo")
> res$Muestra<-as.factor(res$Muestra)
>
> png("Variacion_pi_digitos.png")
> ggplot(data=res,aes(x=Muestra,y=Presicion))+
+   geom_boxplot(fill="skyblue3")+
+   scale_y_continuous(name="Precision")+
+   scale_x_discrete(name="Numero de muestras (10^5)")
>
>
> png("Variacion_pi_diferencia.png")
> ggplot(data=res,aes(x=Muestra,y=Diferencia))+
+   geom_boxplot(fill="steelblue")+
+   scale_y_continuous(name="Diferencia")+
+   scale_x_discrete(name="Numero de muestras (10^5)")
>
> png("Variacion_pi_tiempo.png")
> ggplot(data=res,aes(x=Muestra,y=Tiempo))+
+   geom_boxplot(fill="lightseagreen")+
+   scale_y_continuous(name="Tiempo")+
+   scale_x_discrete(name="Numero de muestras (10^5)")
>
>
> graphics.off()
> rm(list=ls())
> gc()
```

Desarrollo del código

El código R presentado en la sección “Código” para el Reto 1 es resultado de modificaciones (en color negro) al código R desarrollado por Will Kurt [3] con los siguientes objetivos:

1. Activar las librerías “ggplot” y “parallel”, la primera para crear gráficos atractivos mientras que la segunda librería será útil para realizar esta práctica de manera paralela.
2. Modificar rápidamente la cantidad de repeticiones para un experimento con las mismas condiciones con la variable “ciclos”; la variable “varrruns” modifica la cantidad de números al azar que serán necesarios para formar el cuadro y poder realizar los cálculos mencionados en la introducción.
3. Declarar las variables “dat” y “res” como *data.frame* para guardar de manera ordenada los valores π de los diferentes experimentos.
4. Declarar el valor más cercano a la constante π en la variable “rpi”, este valor fue obtenido en línea [4].

5. Substraer los dígitos de la variable “rpi” en la matriz “digpi” para una futura comparación con los diferentes valores obtenidos dentro del experimento.
6. Medir el lapso en el cual se ejecutan: la generación de números al azar para dos dimensiones, la identificación de aquellos puntos que se encuentran dentro un círculo de radio definido y la obtención del valor “mc.pi” que representa el valor obtenido experimentalmente de la constante π.
7. Substraer los dígitos de la variable “mc.pi” en una matriz “digmc” para comparar el valor de cada elemento con el elemento correspondiente de la variable “digpi”.
8. Debido a que la longitud de las matrices “digpi” y “digmc” puede ser distinta, se agregan las filas con valor “0” necesarias para igualar dichas longitudes.
9. Realizar la comparación por elemento de las matrices “digpi” y “digmc” a partir del tercer elemento (únicamente números decimales) hasta encontrar un valor diferente entre ambas matrices.
10. Guardar cuantos decimales son iguales para la constante “rpi” y “mc.pi”y obtener el absoluto de la diferencia aritmética entre ambas.
11. Agregar columnas al *data.frame* “dat” y filas al *data.frame* “res”.
12. Nombrar las columnas del *data.frame* “res”.
13. Crear diferentes gráficos para mostrar los resultados obtenidos.

Resultados

En la figura 4 se muestra el número de decimales que poseen en mismo valor tanto para el valor de π proporcionado en línea [4] y el valor obtenido experimentalmente en esta simulación. Se puede observar que aún y para las muestras más grandes el valor de la media es igual a dos, por lo cual se puede decir que con una muestra pequeña (por ejemplo 100,000 números) es posible obtener el mismo resultado que en muestras más grandes; esto no significa que sólo con una muestra pequeña basta, ya que como también puede observarse una gran cantidad de resultados experimentales sólo coinciden en una decimal, lo que puede conllevar a errores mayores en cálculos posteriores. Se recomendaría utilizar aquellos grupos de muestras que no poseen una gran variación, como los son los grupos de 300,000 y 400,000 números, en los cuales la casi todos sus valores coinciden en al menos 2 decimales.

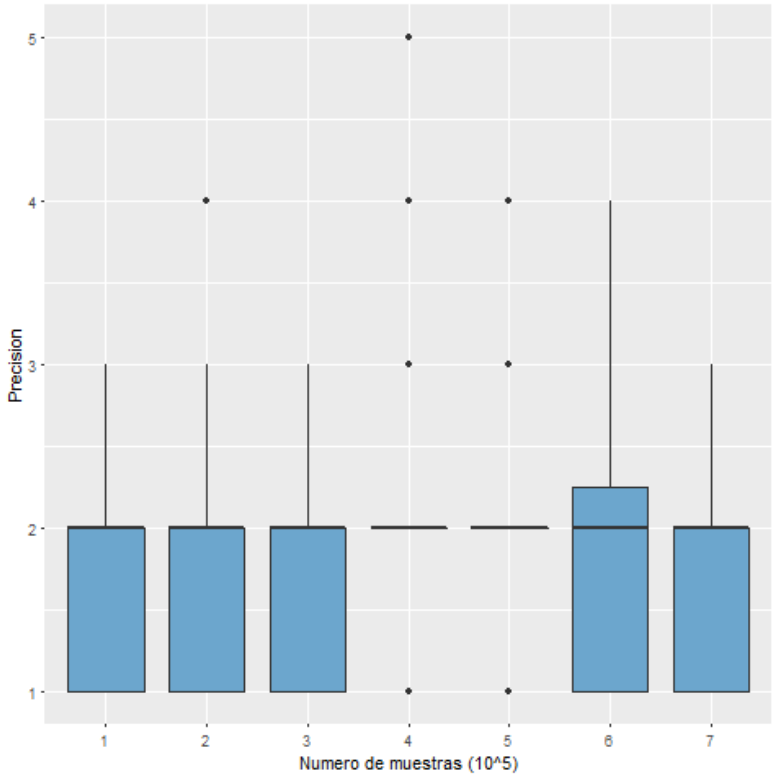

Figura 4 Comparación en la precisión de números decimales iguales para el valor calculado de π a partir de una simulación experimental y el valor más aproximado de π encontrado en línea.

La figura 5 muestra el valor absoluto de la diferencia aritmética entre el valor obtenido experimentalmente para π y la constante obtenida en línea. Se puede observar que el valor de esta diferencia tiende a disminuir conforme el número de muestras aumenta. Esto es coherente con lo esperado, ya que el área el área que ocupan los puntos generados aleatoriamente es mayor y es más cercana al área real, por lo que puede obtenerse con mayor exactitud el valor de π. En esta simulación la diferencia entre ambos valores se encuentra en el orden de las milésimas.

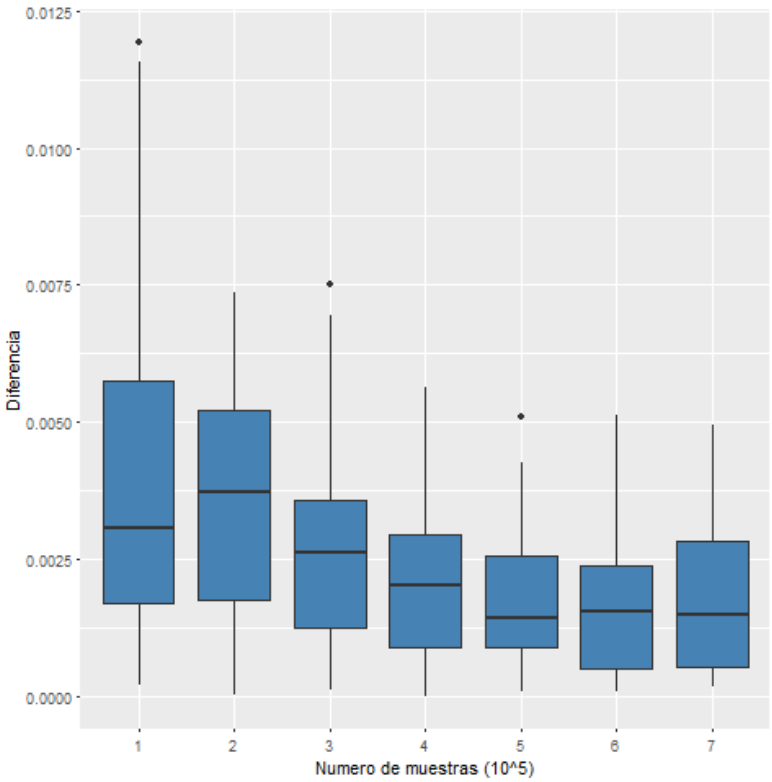

Figura 5 Representación de la diferencia aritmética entre el valor de π obtenido experimentalmente y la constante en línea.

En la figura 6 se muestra el tiempo necesario para calcular π para cada tamaño de muestra. El resultado es consistente con lo que se esperaría, pues a mayor tamaño de muestra, es necesario obtener más números al azar, segregar una mayor cantidad de números y realizar sumas con una mayor cantidad de elementos, lo cual aumenta el tiempo requerido para llevar a cabo todas estas funciones. Se sabe que estas funciones (sumar, separar, generar) son básicas y a pesar de tener 700,000 elementos se realizan casi de manera inmediata pues no toma más de 0.1 segundos para cada operación. Considerando todos los resultaos anteriores, puede ser factibles utilizar tamaños de muestra muy grandes cuando se desee una mayor exactitud. Si sólo se desea una aproximación del valor de π, puede realizarse con una resolución del grado de la milésimas y en relativamente corto tiempo con tamaños de muestra menores.

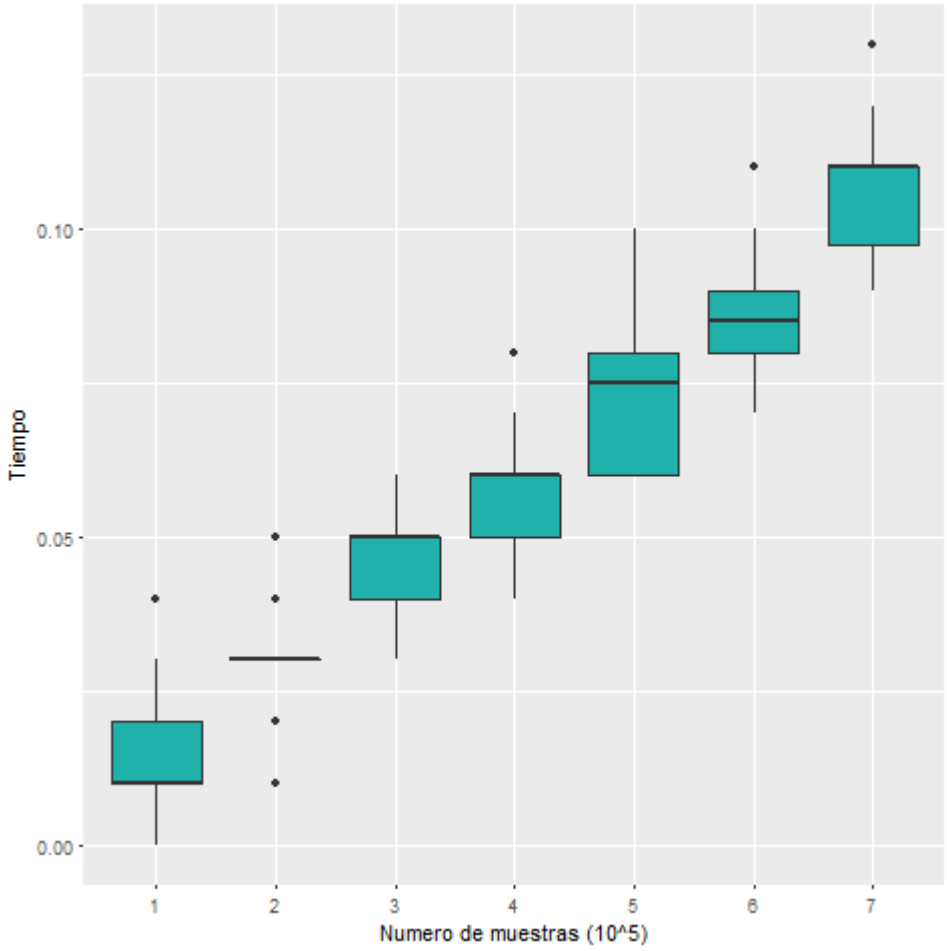

Figura 6 Representación del tiempo tomado por los cálculos del valor de π para diferentes tamaños de muestra.

Referencias

[1] Elisa.dyndns-web.com. (2017). P5 — R paralelo — Schaeffer. [online] Available at: http://elisa.dyndns-web.com/teaching/comp/par/p5.html [Accessed 12 Sep. 2017].

[2] Wolframalpha.com. (2017). Wolfram|Alpha: Computational Knowledge Engine. [online] Available at: https://www.wolframalpha.com/input/?i=integrate++(1%2F(exp(x)%2Bexp(-x)))+from+3+to [Accessed 12 Sep. 2017].

[3] Kurt, W. and Kurt, W. (2017). 6 Neat Tricks with Monte Carlo Simulations. [online] Count Bayesie. Available at: https://www.countbayesie.com/blog/2015/3/3/6-amazing-trick-with-monte-carlo-simulations [Accessed 12 Sep. 2017].

[4] Anon, (2017). [online] Available at: http://www.vaxasoftware.com/doc_edu/mat/numpi15000.pdf [Accessed 12 Sep. 2017].