Práctica No 5: Método Monte-Carlo

Héctor Hernán Montes García

12 de septiembre de 2017

Resumen

En la presente práctica estudiamos tres aplicaciones del método Monte-Carlo, la primera para resolver la integral de una función, la segunda para aproximar numéricamente el número π, y la tercera para construir una función de pronóstico de los nuevos casos del Zika reportados en el estado de Oaxaca-México. Para la primera y segunda práctica evaluamos el impacto del tamaño de la muestra en la exactitud de la aproximación y en los tiempos de ejecución, para el tercer problema calculamos la precisión de los estimados obtenidos de casos de Zika contra los verdaderos valores de casos reportados en el Boletín epidemiológico de la Dirección General de Epidemiología. Asumimos que la serie de casos nuevos de Zika es un proceso estocástico ruido blanco, donde la función de distribución es desconocida pero estimable a partir de pruebas de bondad de ajuste sobre un conjunto de observaciones históricas. Para los dos primeros problemas, los resultados de la experimentación permiten concluir que la precisión aumenta conforme los tamaños de muestra crecen, mientras que los tiempos de ejecución y su variabilidad aumentan para tamaños más grandes de los bloques a paralelizar. Finalmente, para la función de pronóstico de casos del Zika no se obtienen buenos resultados con un modelo que asume ruido blanco, y se sugieren modelos alternativos para trabajos futuros.

1. Planteamiento del problema

En esta práctica estudiaremos tres problemas independientes, a continuación una descripción de cada uno.

1. Aplicar cómputo paralelo para resolver la siguiente integral:

$$\int_3^7 \frac{dx}{e^x + e^{-x}} \tag{1}$$

mediante método Monte-Carlo. Después de algunas consideraciones de cálculo, la integral bajo estudio puede expresarse como dependiente de una función de densidad de probabilidad, de la forma:

$$g(x) = \frac{2}{\pi} \frac{dx}{e^x + e^{-x}} \tag{2}$$

Se aprovecha este hecho para generar números aleatorios bajo dicha distribución usando la librería *distr* de R. Luego evaluamos la proporción

de números que caen en el intervalo de integración $[3, 7]$. Finalmente se usa este resultado para obtener un estimado del área bajo la curva de la función original.

2. Aplicar cómputo paralelo para resolver el problema de aproximación del número π sugerido por Kurt[1]. La aproximación aprovecha nociones geométricas. Se construye una circunferencia inscrita en un cuadrado de lado $2r$ centrado en el origen de un sistema de coordenadas. El centro de la circunferencia también será el origen. Luego se generan pares de números aleatorios bajo una distribución uniforme entre $-r$ y r, para simular las coordenadas horizontal y vertical de un punto interior al cuadrado. Por convención denotaremos tales coordenadas como x y y. La probabilidad de elegir aleatoriamente un punto del cuadrado y que éste caiga dentro del círculo es $\frac{\pi r^2}{4r^2}$, es decir, la razón entre el área favorable al evento y el área total. Podemos ver que el cociente no depende del radio elegido, y este hecho se aprovecha para obtener un estimado de π a partir del estimado de la probabilidad (π es aproximadamente cuatro veces esta probabilidad). Para estimar la probabilidad basta con contar la proporción de puntos cuyas coordenadas pertenecen al interior del círculo, es decir, satisfacen la ecuación $x^2 + y^2 \leq r^2$. Por conveniencia podríamos elegir un cuadrado de lado unitario, es decir, un $r = 0{,}5$.

3. Aplicar el método de Kurt[1] para obtener una estimado de los nuevos casos del Zika en el estado de Oaxaca México, aprovechando la información histórica. Para ello asumimos que el proceso generador de los nuevos casos de Zika es un ruido blanco, y aplicamos métodos de bondad de ajuste para estimar la función de probabilidad subyacente. Para garantizar que los resultados de pronóstico sean discretos, los nuevos casos se modelan como una función de distribución discreta. Una vez obtenida la función de distribución que mejor representa los datos históricos en el sentido de la prueba de bondad de ajuste, esta es usada para generar pronósticos de casos de Zika. La precisión se evalúa usando varias corridas del proceso para pronosticar los restantes casos semanales de Zika que no están considerados dentro de la muestra inicial. Estos casos están reportados en el boletín epidemiológico del año en cuestión. Como cada corrida arroja un pronóstico diferente, se reporta el promedio del error cuadrático medio del pronóstico para una cantidad suficientemente grande de corridas. La evaluación de la calidad del pronóstico se paraleliza para disminuir el tiempo computacional de la experimentación.

2. Resultados

Todas las experimentaciones se han realizado en R, y para el efecto se usó una computadora con sistema operativo Windows 10 Pro de 64bits y procesadorx64 Intel(R) core(TM) i5-4210U, CPU@ 1.70GHz 2.40GHz y memoria instalada de 6.00GB. Una llamada a la función *detectcores(logical=FALSE)* de R, permitió además constatar que la computadora posee dos núcleos físicos disponibles. A continuación se describe los fragmentos más importantes del código usado en cada experimentación, y los resultados numéricos y gráficos obtenidos.

2.1. Resultados para el problema 1

Para evaluar el impacto del tamaño de muestra en la calidad de la aproximación de los valores de la integral, y en los tiempos de ejecución se creó un código en R a partir de ligeras modificaciones al sugerido en clase. Consideramos cuatro tamaños de muestra: $500, 1000, 1500, 2000$, los cuales se encuentran almacenados en el vector *tam*, y sobre cada tamaño de muestra generamos 500 muestras diferentes (este valor es almacenado en el parámetro *cuantos* y permanece fijo a lo largo de toda la experimentación). El experimento procede así:

1. Para un tamaño de muestra prefijado, generamos una muestra particular,y calculamos la cantidad de números aleatorios de la muestra que caen en el intervalo de interés $[3, 7]$.

2. Hacemos lo anterior 499 veces más sobre muestras diferentes de ese mismo tamaño. Los conteos parciales los totalizamos a lo largo de las 500 muestras. Esto nos da el total de números aleatorios que caen dentro del intervalo de interés cuando el tamaño efectivo del experimento es de $500t$ donde t es el tamaño prefijado para cada una de las 500 muestras individuales.

3. Expresamos este total como un porcentaje del tamaño efectivo del experimento, es decir, dividido entre $500t$, y finalmente lo ajustamos por el factor $\frac{\pi}{2}$, para obtener un aproximado del valor de la integral original.

4. Los tres pasos anteriores los repetimos 50 veces, en forma tal que contemos con 50 aproximaciones distintas del valor de la integral para un mismo tamaño de muestra prefijado.

5. Cambiamos el tamaño de muestra por el siguiente en la lista, y repetimos los primeros cuatro pasos hasta agotar todos los tamaños de interés.

El siguiente fragmento de código muestra la lógica del experimento. Note que la paralelización se hace al nivel de las 500 muestras de un mismo tamaño de muestra, y no se paraleliza ni las 50 réplicas, ni los cambios de tamaño. Los datos de tiempos de ejecución y los valores de la integral se guardan en el *data.frame* llamado *resultados*.

```
1  for (t in tam)
2  {
3     clusterExport(cluster,"t")
4     for (i in 1:50)
5    {
6        clusterExport(cluster,"i")
7        t0<-Sys.time()
8        integrales[i]<-sum(parSapply(cluster,1:cuantos,function(x)
             {parte(desde,hasta,generador(t))}))*(pi/(2*t*cuantos))
9        tf<-Sys.time()
10       tiempos[i]=(tf-t0)
11   }
12   resultados<-rbind(resultados,cbind(Tamano=t,Integral=
          integrales,Tiempo=tiempos))
13 }
```

Para la parte de análisis gráfico, se usó la librería *ggplot2* la cual ofrece poderosas herramientas de manipulación gráfica. También se hizo uso de la librería *gridExtra* para la generación de varios gráficos en un mismo dispositivo. El código usado para graficar es el siguiente:

```
1  Valor_referencia<-atan(exp(7))-atan(exp(3))
2  png("aproximaciones_y_errores.png",width = 617, height = 354)
3  plot1<-ggplot(resultados, aes(x=as.factor(Tamano),
4                          y=Integral,
5                          fill=as.factor(Tamano)),xlab(Tamano))+geom_
                               boxplot(alpha=0.3)+theme(legend.position
                               ="none")+geom_hline(yintercept = Valor_
                               referencia)+labs(x = "Tama\u{F1}o_de_
                               muestra",y = "Valores_aproximados")
6  plot2<-ggplot(resultados, aes(x=as.factor(Tamano),
7                          y=abs(Integral-Valor_referencia)/Valor_
                               referencia, fill=as.factor(Tamano)),xlab
                               (Tamano))+geom_boxplot(alpha=0.3)+theme(
                               legend.position="none")+labs(x = "Tama\u
                               {F1}o_de_muestra",y = "Error_relativo")
8  grid.arrange(plot1, plot2, nrow=1, ncol=2)
9  graphics.off()
10 png("tiempos.png")
11 ggplot(resultados, aes(x=as.factor(Tamano),y=Tiempo, fill=as.
       factor(Tamano)),xlab(Tamano))+geom_boxplot(alpha=0.3)+theme(
       legend.position="none")+labs(x = "Tama\u{F1}o_de_muestra",y
       = "Tiempos_computacionales")
```

Note que para el cálculo de los errores de aproximación de la integral se está usando el valor teórico, dado que el integrando cuenta con antiderivada conocida y no es necesario proceder a integración numérica, pues:

$$\int_3^7 \frac{dx}{e^x + e^{-x}} = \int_3^7 \frac{e^x dx}{e^{2x} + 1} = \int_{e^3}^{e^7} \frac{du}{u^2 + 1} = \arctan(u)\Big|_{e^3}^{e^7} \tag{3}$$

Por lo tanto, hemos usado en la línea uno del código precisamente la sentencia que evalúa esta expresión. Con base en dicho valor de referencia se ha calculado el error relativo de aproximación. Observe que una línea de referencia en el valor teórico es ubicada en la gráfica de valores aproximados. Esto con el fin de visualizar más fácilmente cómo aumenta la precisión conforme el tamaño de muestra crece. Finalmente los resultados gráficos se ofrecen en las figuras 1 y 2.

Por último vienen las pruebas estadísticas, es decir, la manera formal como nos preguntamos si las diferencias detectadas gráficamente son estadísticamente significativas o sólo se deben al azar. Para ello aplicamos pruebas no paramétricas para la comparación de muestras independientes, las cuáles no requieren del cumplimiento del supuesto de normalidad sobre los datos de partida. Una prueba ampliamente recomendada para este caso es la prueba de Kruskal Wallis para la comparación de k muestras independientes. Siguiendo a Conover[2], es importante recordar los supuestos sobre los que se basa la prueba para una adecuada interpretación.

1. Todas las muestras son aleatorias y provienen de sus respectivas poblaciones.

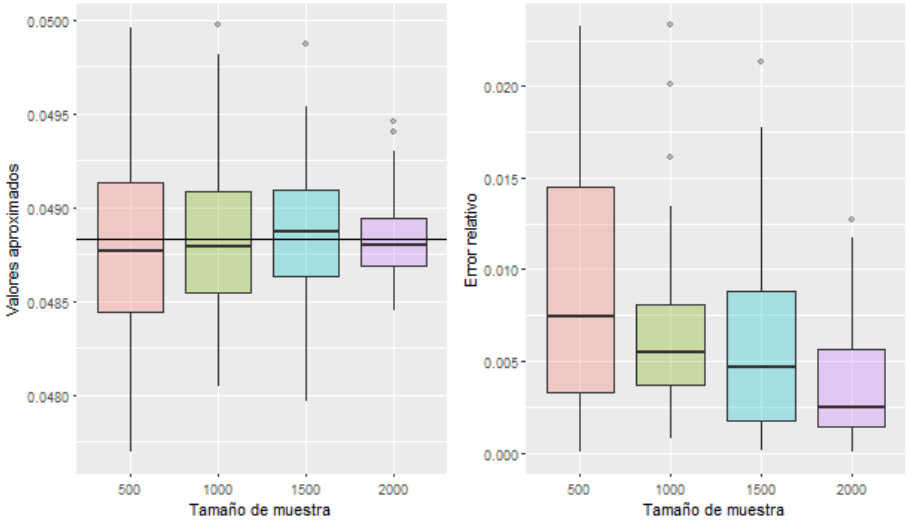

Figura 1: Impacto del tamaño de muestra en los valores aproximados y los errores relativos

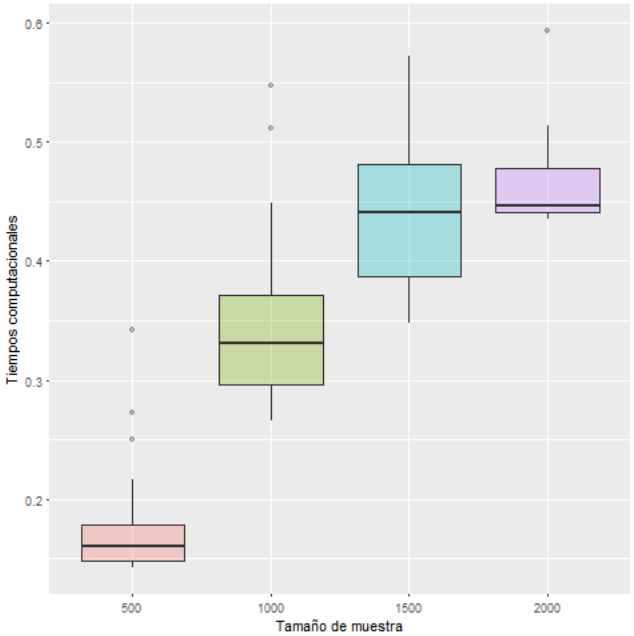

Figura 2: Impacto del tamaño de muestra en los tiempos de ejecución

2. Adicional a la independencia de las observaciones dentro de cada una de las muestras, se debe respetar la mutua independencia de los datos entre las muestras.

3. La escala de medida de las variables de interés es al menos ordinal.

4. O bien las k funciones de distribución poblacional son iguales, o al menos una tiende a dar valores más grandes que alguna de las demás.

Todos estos supuestos se cumplen en nuestra experimentación debido al procedimiento de muestreo usado que garantiza tanto la independencia intra-muestral como inter-muestral, y la pertenencia de las observaciones a las poblaciones de interés. Evidentemente nuestras variables son continuas así que también se cumple el requisito aplicado a la escala, y por último, dado que estamos interesados en detectar posibles incrementos en los tiempos de ejecución o decrementos en los niveles de error para cuando los tamaños de muestra aumentan, la prueba es sensible a nuestra hipótesis en juego, las cuáles son:

H_0: *Todas las k funciones de distribución poblacional son idénticas.*
H_a : *Al menos una de las poblaciones tiende a arrojar observaciones más grandes que una cualquiera de las demás poblaciones.*

Ya que la prueba de Kruskal- Wallis ha sido diseñada para ser sensible a la diferencia entre las medias de las k poblaciones, esta hipótesis alternativa algunas veces es también establecida como:

H_a : *Las k poblaciones no tienen medias idénticas.*

En nuestro contexto de investigación las hipótesis para el caso de análisis de errores de aproximación recaerán sobre las medias de los tiempos de ejecución y la media de los errores relativos. R nos permite ejecutar pruebas de Kruskall-Wallis sobre los datos de la experimentación, con bastante facilidad:

```
1  kruskal.test(abs(resultados$Integral-Valor_referencia)/Valor_
       referencia~as.factor(resultados$Tamano))
2 kruskal.test(resultados$Tiempo~as.factor(resultados$Tamano))
```

La línea 1 ejecuta la prueba para los errores relativos, mientras que la línea 2 lo hace para los tiempos de ejecución. Los resultados del análisis se presentan en la Tabla 1. Cabe apuntar que la tabla (pese a lo sencilla) fue construida con ayuda de la librería *xtable* de R, la cual se recomienda ampliamente para la generación de código latex para aquellas tablas construidas en R que requieren ser trasladadas a un documento Latex.

Tabla 1: Prueba de igualdad de medias para errores relativos y tiempos de ejecución en función del tamaño de la muestra

	Estadístico	Grados de libertad	Valor p
Error	20.73	3	0.00012
Tiempo de ejecución	153.24	3	0.00000

2.2. Resultados para el problema 2

Para este problema se procedió de forma muy similar al problema 1, excepto que los tamaños de muestra considerados fueron un poco más grandes con valores de $5000, 10000, 15000, 20000$ [1]. Como se había mencionado antes, y según

[1]Se comprende la necesidad de ampliar el presente estudio a tamaños de muestra más grandes, en especial para el problema 1, pero las limitaciones tecnológicas asociadas al equipo usado para la experimentación, impidieron un estudio más exhaustivo.

la propuesta de Kurt[1], el problema de hallar una aproximación de π puede abordarse por método Monte-Carlo generando puntos aleatorios uniformemente distribuidos sobre un cuadrado de lado unitario, y contando la cantidad de dichos puntos que caen dentro del círculo de radio 0,5 con centro en el centro mismo del cuadrado. La figura 3 muestra la forma en que trabaja el método de Kurt, al realizar el conteo de la proporción de puntos que caen dentro de la circunferencia (puntos rojos) respecto al total de puntos generados (puntos rojos + azules).

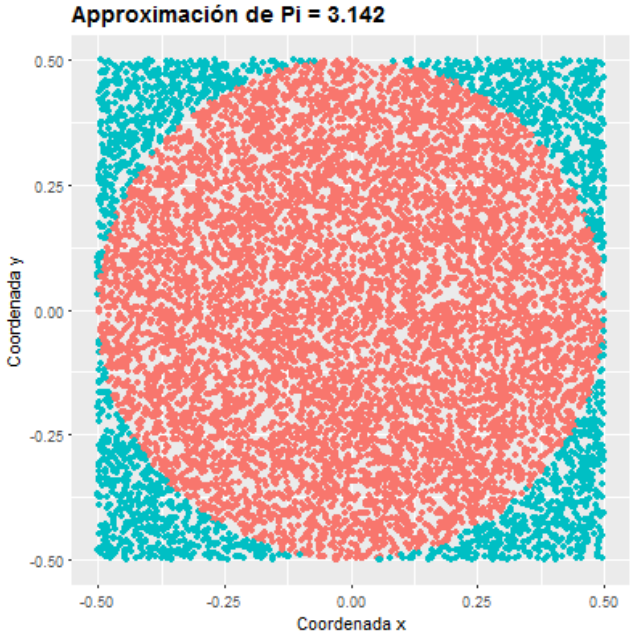

Figura 3: Aproximación de PI por método Monte-Carlo

Para lograr una gráfica un poco más estética, se utilizó la siguiente instrucción en el lenguaje de gramática de gráficos disponible en la librería *ggplot2*

```
1 png("Grafico_pi.png",width = 400,height = 400)
2 ggplot(datos,aes(x,y),pch='.',asp=1, main ="Aproximaci\u{F3}n_de
      _PI", xlim=c(-0.5,0.5),ylim=c(-0.5,0.5))+geom_point(aes(col=
      ifelse(in.circle,"blue","grey")))+xlab("Coordenada_x")+ylab(
      "Coordenada_y")+ggtitle(paste("Approximaci\u{F3}n_de_Pi_=",
      mc.pi))+theme(plot.title = element_text(size=rel(1.3),face="
      bold",color="Black", lineheight=1.5),legend.justification =
      "top",legend.position="none")
3 graphics.off()
```

La implementación de la experimentación para la aproximación de π siguió los siguientes pasos generales:

1. Para una cantidad de puntos prefijada (tamaño de muestra) calcular la cantidad de ellos que caen dentro de la circunferencia usando una función de conteo convenientemente diseñada.

2. Realizar el proceso anterior 499 veces adicionales, para un total de 500 conteos.

3. Totalizar los conteos a lo largo de las 500 muestras de tal forma que se conforme una muestra efectiva de tamaño $550t$, donde t es la cantidad de puntos prefijada para cada lote de puntos.

4. Expresamos el total como una proporción del tamaño de muestra efectivo, es decir, divididos entre $500t$, y cuadruplicamos este total con el fin de obtener el estimado de π deseado.

5. Repetimos 50 veces los pasos anteriores para contar con varias estimaciones de π.

6. Cambiamos el tamaño de muestra por el siguiente en la lista, y repetimos los primeros cuatro pasos hasta agotar todos los tamaños de interés.

7. Construimos salidas gráficas y tablas con el resumen de los resultados obtenidos, las cuales se muestran en las figuras 4 y 5, y en la Tabla 2

La estructura del código usado para esta experimentación es esencialmente igual a la usada en el problema 1, excepto por la función implicada en la paralelización:

```
f <- function(runs){
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
return((sum(in.circle)))
}
aproximado[i]<-sum(parSapply(
    cluster,1:cuantos,function(x) {f(t)}))* (4/(t*cuantos))
```

La cual es pasada después como argumento al ciclo parSapply, así:

```
aproximado[i]<-sum(parSapply(
    cluster,1:cuantos,function(x) {f(t)}))* (4/(t*cuantos))
```

Detalles adicionales se pueden leer en el código disponible en el repositorio.

Tabla 2: Prueba de igualdad de medias para errores relativos y tiempos de ejecución de las aproximaciones de PI

	Estadístico	Grados de libertad	Valor p
Error	24.60	3	0.00002
Tiempo de ejecución	184.42	3	0.00000

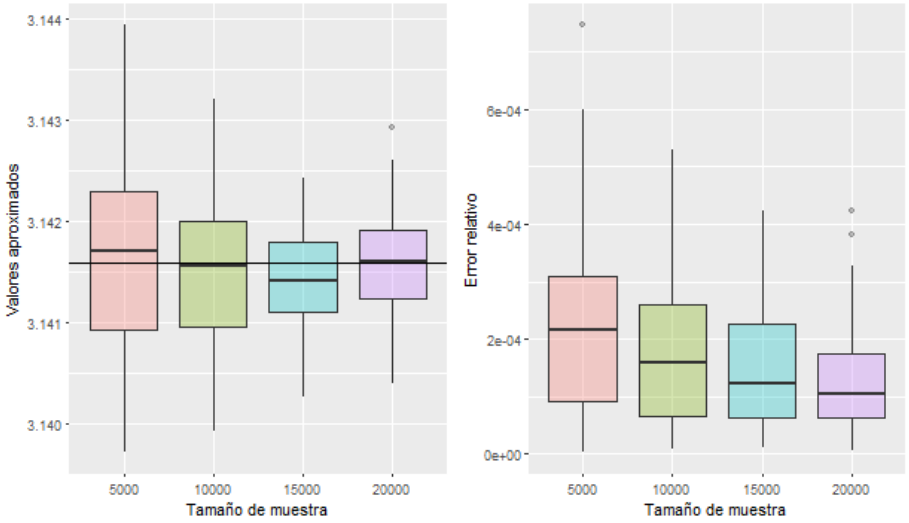

Figura 4: Impacto del tamaño de muestra en la precisión de las aproximaciones de PI

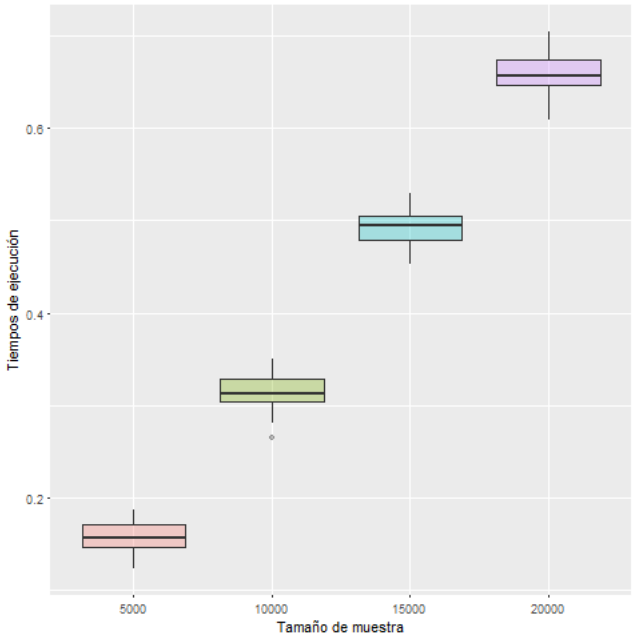

Figura 5: Impacto del tamaño de muestra en los tiempos de ejecución de las aproximaciones de PI

3. Conclusiones

Tanto para el problema de aproximación de la integral como para el problema de aproximación de PI se notan incrementos en los niveles de precisión, esto es, disminución del error relativo, y de la dispersión de las estimaciones conforme el tamaño de muestra aumenta. Lo anterior se encuentra ratificado no sólo por

los resultados gráficos sino por los valores p tan pequeños obtenidos en las pruebas de Kruskall-Wallis, los cuales estarían siendo indicativos de rechazo de la hipótesis de tiempos y niveles de error relativo promedio iguales entre los diferentes tamaños de muestra.

Esto significa que aumentos en precisión, vienen acompañados de mayor costo computacional, pues los tiempos de ejecución aumentan también conforme se requieren muestras de tamaño más grande. El practicante tendrá por tanto que encontrar la mejor solución de compromiso entre los niveles de precisión deseados y el costo computacional que se está dispuesto a asumir. Esto se podría lograr calculando cuánto aumenta el costo computacional por unidad de aumento en la precisión deseada (por ejemplo a través de un modelo de regresión entre tiempos vs precisión). No obstante para estos dos problemas, los tiempos de ejecución no son exageradamente elevados, aun para los tamaños de muestra más grandes, así que queda demostrada la potencia de los métodos Monte-Carlo para solucionar problemas que se adecuan al enfoque de simulación estocástica.

Referencias

[1] Kurt, W. *6 Neat Tricks with Monte Carlo Simulations — Count Bayesie*, Probably a probability blog, March 24, 2015. https://www.countbayesie.com/blog/2015/3/3/6-amazing-trick-with-monte-carlo-simulations

[2] Conover, W.J. (1980) *Practical nonparametric statistics.* Wiley New York, pag 289-290.