Tarea 5
Simulación de Sistemas

Beatriz Alejandra García Ramos

A 12 de Septiembre de 2017

1. Método Monte-Carlo

Aunque se tomen muestras pequeñas para ésta práctica el Método de Monte-Carlo nos ayuda a encontrar de manera muy aproximada distribuciones o valores que no se conocen y son difíciles de calcular de forma analítica.
Will Kurt tiene simulaciones del método Monte-Carlo, como encontrar el valor aproximado de π que se utiliza para el primer reto de la tarea, y encontrar el valor de una integral con una función complicada de trabajar analíticamente. En base a ello nos podemos dar una idea sobre cómo se trabaja éste método en un código de R.

2. Tareas

- Examinar la precisión que se tiene en el valor de la integral cuando se hacen variaciones en el tamaño de muestra, comparar la aproximación del programa con la aproximación que tiene Wolfram Alpha y medir el tiempo de ejecución.
- Examinar la precisión del valor de π de Kurt paralelizando el código, comparar la aproximación de éste con la aproximación de Wolfram Alpha y medir el tiempo de ejecución.
- Examinar los datos que se tienen sobre los reportes de Zika en el estado de Oaxaca durante las primeras treinta y cuatro semanas haciendo comparaciones paralelizando también la técnica de Kurt y ver la precisión de los resultados.

3. Solución

Para examinar las aproximaciones de los valores no se necesita saber qué está sucediendo con las imágenes que se tenían en el código original así que se puede eliminar esa parte que las determinaba. En cambio, para poder tener una buena visión sobre lo que sucede con las distintas aproximaciones dependiendo de las variaciones de los tamaños de muestra se crearon gráficos que nos muestran qué tan cerca se encuentran los valores obtenidos del código con respecto al valor aproximado de Wolfram Alpha, así que en los gráficos se tiene una línea horizontal roja marcada para indicar el valor que tiene Wolfram Alpha y puntos para indicar los valores aproximados del código. Ambos códigos tienen como medición del tiempo de ejecución la función `Sys.time`() de R, `Sys.time`() de inicio antes de `foreach` y `Sys.time`() de finalización después de éste.

Las modificaciones de los códigos se pueden encontrar en GitHub.

3.1. Tarea base

Para la tarea base se debe cambiar la manera en que se toma la muestra. Entonces, se crea un `for` que permite tomar los distintos tamaños de la variable `muestra` y un `for` que permite hacer repeticiones de cada tamaño de muestra. Una vez que se entra en ellos se mide el tiempo de ejecución de la variable

`montecarlo`, y se calcula el valor de la integral. Éstos datos se guardan en un `data.frame()` donde la primer columna es el valor aproximado de la integral y la segunda es el tiempo de ejecución.

Los datos obtenidos de la aproximación se hicieron con siete muestras distintas y cinco replicas para cada una de ellas. Como se observa en la figura 3.1 en un principio las aproximaciones no se acercaban al valor que tiene Wolfram Alpha para la integral pero conforme el tamaño de muestra va aumentando las aproximaciones son más parecidas a dicho valor.

Esto sucede porque al tomar mayor cantidad de muestra estamos tomando más espacio debajo de la función que se está integrando y es como si hubiera menos huecos en ésta área, lo cual nos dará más precisión sobre el valor de la integral, es por eso que podemos obtener un resultado muy similar al que se tiene en Wolfram Alpha.

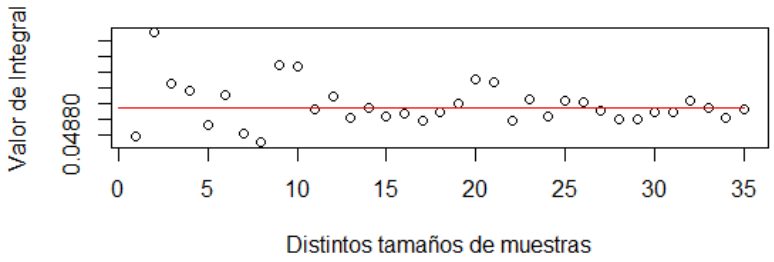

Figura 1: Aproximaciones del valor de la integral comparadas con el aproximado de Wolfram Alpha.

Como se puede observar en el cuadro 3.1 tenemos los distintos tiempos de ejecución dependientes del tamaño de muestra que se le asigna a nuestro código en cada fase que se realiza. Como es de esperarse, se tiene que mientras va aumentando el tamaño le toma más tiempo al código hacer las funciones que se le solicitan.

Los datos están dados en segundos, excepto la última columna que toma minutos, así que las variaciones que se tienen dependiendo del tamaño de muestra son proporcionales a éste, y es notorio también que en cada tamaño los tiempos son casi iguales, eso quiere decir que no importa cuántas réplicas se realicen para cada muestra, éstas siempre tendrán un tiempo establecido dependiente del tamaño.

Podemos observar también que los tiempos registrados no son grandes, así que el tiempo de ejecución se puede considerar como bueno, dependiendo de la cantidad de repeticiones que queremos que se haga es por lo que podría tardarse.

Tamaños						
500	1000	2500	5000	9000	15000	36000
5.732624	3.295606	8.196015	16.199031	29.212053	48.869489	1.932727
1.718003	3.325806	8.151215	16.207230	29.427053	48.988487	1.943870
1.685803	3.266400	8.261815	16.210231	29.711254	48.664689	1.956334
1.686803	3.341406	8.198416	16.225632	29.224055	49.245889	1.957404
1.654603	3.260406	8.138016	16.239633	28.992252	48.774488	1.957737

Cuadro 1: Tiempos de ejecución de distintas muestras.

3.2. Reto 1

Para el reto lo que se hizo fue utilizar la manera en que Kurt encuentra aproximaciones para el valor de π y ésto se utilizó como la función del código, la cual se manda llamar cuando se quiere utilizar la paralelización con `foreach`. De igual manera, como en la tarea base, se utilizan distintos tamaños

de muestra, se hacen replicas de cada uno de éstos y se considera el tiempo de ejecución para cada tamaño.

En la figura 3.2 tenemos las aproximaciones del valor π con doce tamaños distintos y veinte repeticiones de cada uno de los tamaños.

El valor aproximado que nos da Wolfram Alpha es el indicado con una línea horizontal roja y podemos notar que entre más crece el tamaño de la muestra que se toma más cerca están los valores de las aproximaciones al valor de π según Wolfram, lo cual nos permite decir con mayor seguridad que alguno o algunos de ellos serán exactamente ese valor aproximado de π.

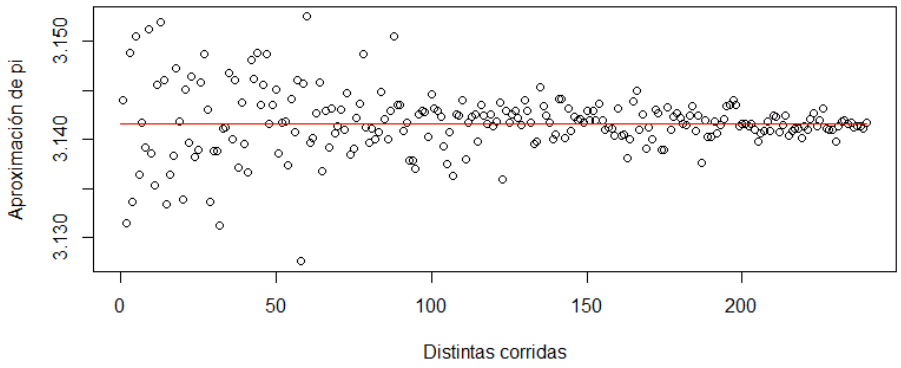

Figura 2: Aproximaciones del valor de π comparadas con el aproximado de Wolfram Alpha.

Dado que los tiempos de ejecución que se obtienen de éstas aproximaciones son muy pequeños y son muchas las observaciones que se hicieron para examinar la función del código se encuentran los valores de los tiempos en la figura 3.2, en la cual tenemos que éstos van en incremento dependiendo del tamaño de muestra que se tiene, lo que es correcto ya que entre más grande es éste tamaño más debería tardar en ejecutarse. Se observa además que los tiempos son constantes en las distintas repeticiones para cada distinto tamaño, también se puede ver que no excede los diez segundos de ejecución por lo cual estamos completamente seguros que a pesar de poner una muestra de tamaño elevado al millón nuestro código no tendrá problemas en ejecutarlo de manera rápida y precisa.

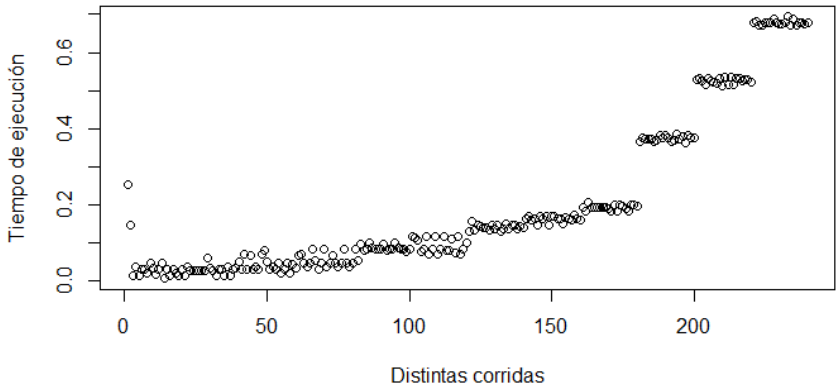

Figura 3: Tiempos de ejecución de los valores aproximados de π