Entrega 6

Augusto Souto

15 de mayo de 2019

Índice

1. Ejercicio 10.1:

1.1. Consigna

Resumir en un texto (de entre media y una carilla) los principales contenidos del paper de P. L'Ecuyer, "Software for Uniform Random Number Generation: Distinguishing the Good and the Bad", Proceedings of the 2001 Winter Simulation Conference, IEEE Press, Dec. 2001, 95-105. ¿Cuál es el objetivo del trabajo? ¿Qué generadores de números seudo-aleatorios discute? ¿Cuáles son los hallazgos de esta investigación, y cuáles las conclusiones y recomendaciones presentadas por el autor?

1.2. Resumen

El trabajo tiene como objetivo principal introducir al lector al testeo de numeros aleatorios a través de la libreria TestU01, escrita en lenguaje C, cuyo fin es la generación y testeo de números aleatorios. Tal libreria está organizada en 4 clases principales de modulos que sirven a tales fines:

- Aquellas que implementan generadores de numeros aleatorios (RNGs).
- Aquellas que implementan tests estadísticos para numeros aleatorios.
- Aquellas que implementan baterías predefinidas de tests para numeros aleatorios.
- Aquellas que implementan herramientas para familias enteras de generadores de numeros aleatorios.

Al principio del documento, en las secciones 3, 4 y 5, se discuten los criterios de calidad que deben tener los numeros aleatorios, explicando diferentes tests estadísticos que sirven a tal fin. Estos tests se dividen en aquellos destinados al testeo de numeros aleatorios ubicados en el intervalo unitario (entre 0 y 1) y aquellos destinados al testeo de bits aleatorios (unos y ceros). Las estrategias de aplicación de los mismos se separan entre aquellas dirigidas a una secuencia entera de n numeros o bits y aquellas dirigidas a n secuencias de largo t.

Posteriormente, la sección 6 introduce los contenidos de la libreria, que està compuesta de modulos cuyas clases fueron ya mencionadas.

Finalmente, se testean algunos de los generadores de numeros aleatorios mas conocidos mediante la aplicación de la libreria y se hacen conclusiones en las secciones 7 y 8.

Entre los generadores testeados y discutidos se incluyen aquellos que pertenecen a la familia de los generadores lineales congruenciales (LCGs), LCSs combinados, generadores recursivos multiples (MRGs), basados en decimación, con rezagos de Fibonacci, SBW (*substract with borrow*) , combinados mixtos, LFSR y GFSR, inversivos y otros generadores no lineales. A estas clases de generadores pertenecen los generadores que usan algunos softwares de uso masivo, como Matlab (mixtos combinados) o Excel (LCGs combinados).

Los tests estadísticos(los cuales consideran un p-valor de 10^{-10}), hechos a través de los paquetes de pruebas "Small Crush", "Crush" y "Big Crush" de TestU01, muestran que, para muchos de los generadores testeados, se rechaza la hipótesis nula de aleatoriedad de los numeros generados. En particular, se halla que los generadores muchos de estos programas, como Excel, Matlab, R, etc) rechazan la hipótesis nula de que sus numeros son aleatorios. No obstante, algunos tipos de generadores, como, por ejemplo, algunos que están basados en rezagos de Fibonacci o en SWB no rechazan la hipótesis nula.

Por último, el autor recomienda poner más atención en los generadores de numeros aleatorios que combinan elementos de diferentes familias, dado que tienen propiedades deseables. Por ejemplo, se puede demostrar teóricamente su uniformidad.

2. Ejercicio 11.1

Para generar un punto aleatorio (X1, X2) en un círculo de centro (0, 0) y radio 1, es posible hacerlo de la forma siguiente (derivación disponible en las páginas 234 y 235 del libro de referencia del curso, "Monte Carlo: concepts, algorithms and applications", Fishman 1996):

- Se genera un valor aleatorio r, de distribución $F_r(x) = x^2$ para $0 \leq x \leq 1$, y 0 para cualquier otro x;

- Se generan dos v.a. independientes Z_1 y Z_2 de distribución normal (0, 1);
- Se calcula $X1 = \frac{rZ_1}{\sqrt{z_1^2+z_2^2}}$ y $X2 = \frac{rZ_2}{\sqrt{z_1^2+z_2^2}}$ Utilizar esta propiedad para volver a resolver el Ejercicio 6.1 parte a: estimar la integral de la función $\frac{200xy}{max(x,y)}$ sobre la región definida por el círculo con centro en $(05, 05)$ y radio 0.4, en base a 106 replicaciones, pero generando únicamente valores de puntos dentro del círculo.

Comparar la precisión obtenida con la alcanzada en el ejercicio 6.1.

2.1. Resolución

Se implementa la siguiente función para lograr la estimación numérica mediante usando Lebesgue-Stieltjes:

```
lebesgue_stieltjes<-function(n_sim, radio ){

n<-n_sim

r<-radio
set.seed(10)

x<-runif(n,0,r^2)
rad<-x^0.5

z_1<-rnorm(n,0,1)
z_2<-rnorm(n,0,1)

x<-(rad*z_1)/( ((z_1^2) +(z_2^2) )^0.5) %>% as.matrix()
x<-x+0.5

y<-(rad*z_2)/( ( (z_1^2) +(z_2^2) )^0.5) %>% as.matrix()
y<-y+0.5

coor<-cbind(x, y)

#centro#
c_x<-rep(0.5,  n)
c_y<-rep(0.5,  n)
cent<-cbind(c_x, c_y)

area<-(r^2)*pi #area del circulo
valor<-vector("numeric", length=n)
#valores que toma la función en los puntos sorteados dentro de la region#
t1<-Sys.time()
for (i in 1:n) {
      valor[i]=((200*x[i]*y[i])/max(x[i],y[i]))*area #ajusto por densidad con el area
}


s<-vector("numeric", length=n)
t<-vector("numeric", length=n)

t[1]=0
s[1]=valor[1]

for (i in 2:n) {
  t[i]<-t[i-1]+(1-(1/i))*(valor[i] -( s[i-1] / (i-1)) )^2
  s[i]<-s[i-1]+valor[i]
  }
```

```
int<-(s[n]/n) #integral por Lebesgue Stieltjes

var_int<-t[n]/(n-1) #VARIANZA ESTIMADA#
var_est<-var_int/n #VARIANZA MUESTRA

z2<-qnorm(0.975)^2
epsilon<-sqrt(n/(z2*var_int))

epsilone<-sqrt((z2*var_int)/n)

n_es<-(z2*var_int)*10^4

inf<-int-qnorm(0.975)*(var_int/n_es)^(1/2)
sup<-int+qnorm(0.975)*(var_int/n_es)^(1/2)




t2<-Sys.time()
print(t2-t1)


options("scipen"=999, "digits"=4)

return(c(int, var_est, inf, sup))



}
```

La estimación numérica arroja el siguiente resultado. El valor del mismo es muy similar al del ejercicio 3.1, ya que la estimación en dicho caso fue de 38.21435, presentandose una diferencia recién en el tercer numero decimal. También se puede observar que, para el mismo nivel de confianza (95 %), el intervalo de confianza se reduce significativamente desde [38.114,38.314] hasta [38.207,38.227].

```
lebesgue_stieltjes(n_sim = 10^6, radio = 0.4)
```

```
## Time difference of 1.955111 secs
```

```
## [1] 38.2169775  0.0002583 38.2069775 38.2269775
```

3. Ejercicio 11.2

En algunas aplicaciones (por ejemplo cálculo de volúmenes de troncos), se modela un sólido mediante una aproximación geométrica.

En particular, consideraremos un sólido generado por la rotación de una curva $y = y(x)$ en torno a un eje. Tomaremos la familia de curvas para $x \in [0, H]$, donde R es el radio en la base, H la altura máxima considerada, y r un factor de forma (r = 1 corresponde a un paraboloide, r = 2 a un cono, etc.). Es decir, para cada x hay un radio $y(x)$, y por rotación a esa altura se engendra un círculo de área $y(x)^2$Ö??. El volumen total (para R y H fijos) es la integral en x de 0 a H de esas áreas.

Supondremos que la altura H es aleatoria y verifica $H = H0 + K$, donde H_0 es una altura mínima y K sigue una distribución lognormal de media K_m y varianza K_v. En este caso, el volumen promedio es la integral del volumen

para cada valor H ponderado por la funci´on de densidad de H.

- Dar el seudocódigo de un procedimiento Monte Carlo para estimar el volumen promedio y el diámetro promedio de un sólido de estas características, con parámetros de entrada R, H_0, K_m, K_v, y r.
- Escribir el programa correspondiente.
- Realizar cálculos para estimar los valores mencionados (volumen y diámetro promedio), con $R = 19$cm, $H_0 = 10m$, $H_m = 5m$, $H_v = 4m^2$, y tres valores distintos de r: $1, \frac{4}{3}$ y 2.

3.1. Resolución

3.1.1. Pseudocodigo

```
Set parameters R, H_0, K_m, K_v and r
Set parameter n

define function estimar with arguments  R, H_0, K_m, K_v, r and n

  define k distributed log-normal with log mean K_m and log sd K_v

  set seed

  Draw n k samples

  altura=H_0+k

  radio=R*sqrt((x/max(x))^r)

  diametro=radio*2

  densidad=log normal density with log mean K_m and log sd K_v  of altura

  area=pi^2 *densidad* pi

  volumen=sum of elements of vector area

return volumen and mean diametro
```

3.1.2. Código

A continuación se presenta el còdigo utilizado.

```
estimar<-function(r, k_m, k_v, Hmin, R, n){


set.seed(10)
t1<-Sys.time()
k<-rlnorm(n,meanlog = log(k_m), sdlog=log(k_v^0.5) ) %>% as.matrix()
x<-Hmin+k #altura#

H<-max(x) #maxima altura#

radio<-R*(((x/H)^r)^0.5) #radio a una altura determinada#

diametro<-radio*2
```

```
dens<-dlnorm(x ,meanlog = log(k_m), sdlog=log(k_v^0.5) ) #D F(x) #D F(x)

area<-dens*pi*radio^2 #area en una altura#

volumen<-sum(area)

diam<-sum(diametro*dens)/sum(dens)
t2<-Sys.time()

print(t2-t1)
#return(c(volumen, diam))
return(cat("El volumen estimado es de ",volumen,"metros cuadrados y el diametro medio es", diam, "metros"))

}
```

3.1.3. Cálculos

A continuación se presentan los cálculos para los diferentes valores del parámetro de forma (r). Como es de esperar, dado que el valor 1 del parámetro corresponde a un paraboloide y el 2 a un cono (más angosto a medida que aumenta la altura), al aumentar el parámetro el volumen se reduce.

```
estimar(r=1,k_m=5, k_v= 4^2, Hmin = 10, R=0.19, n=10^6)
```

```
## Time difference of 0.796 secs
## El volumen estimado es de  4.391 metros cuadrados y el diametro medio es 0.02026 metros
```

```
estimar(r=4/3,k_m=5, k_v= 4^2, Hmin = 10, R=0.19, n=10^6)
```

```
## Time difference of 0.795 secs
## El volumen estimado es de  0.6456 metros cuadrados y el diametro medio es 0.007669 metros
```

```
estimar(r=2,k_m=5, k_v= 4^2, Hmin = 10, R=0.19, n=10^6)
```

```
## Time difference of 0.785 secs
## El volumen estimado es de  0.0149 metros cuadrados y el diametro medio es 0.00111 metros
```

4. Información sobre la Implementación de los Ejercicios

Se utiliza el lenguaje R para computar la solución, en un equipo intel core i 3 M 380 con 2.53 GHz y 4 Gb de memoria.