

# Simulación - Segunda tarea

Sergio Arnaud Gómez 159189

10 de septiembre del 2018

1. Probar por inducción que para un GLC:

$$Z_k \equiv \left[ a^k Z_0 + c \frac{a^k - 1}{a - 1} \right] \text{mod } m$$

Demostración: (Por inducción sobre k)

(Base de inducción) si  $k = 0$  tenemos que:

$$\begin{aligned} a^k Z_0 + c \frac{a^k - 1}{a - 1} &= a^0 Z_0 + c \frac{a^0 - 1}{a - 1} \\ &= Z_0 + c \frac{1 - 1}{a - 1} \\ &= Z_0 \\ &\equiv Z_0 \text{mod } m \end{aligned}$$

(Hipótesis de inducción) Ahora supongamos que el resultado válido para  $k = n$  y probemos la afirmación para  $n + 1$ .

Por un lado, por la definición de los generadores lineales congruenciales tendemos que :

$$Z_{n+1} \equiv (aZ_n + c) \text{mod } m \quad (1)$$

Por otro lado, por la hipótesis de inducción tenemos que:

$$Z_n \equiv \left[ a^n Z_0 + c \frac{a^n - 1}{a - 1} \right] \text{mod } m$$

Trabajando con esta última expresión obtenemos:

$$\begin{aligned}
& Z_n \equiv \left[ a^n Z_0 + c \frac{a^n - 1}{a - 1} \right] \text{mod } m \\
\Rightarrow & aZ_n \equiv a \left[ a^n Z_0 + c \frac{a^n - 1}{a - 1} \right] \text{mod } m \\
\Rightarrow & aZ_n + c \equiv a \left[ a^n Z_0 + c \frac{a^n - 1}{a - 1} \right] + c \text{mod } m \\
\Leftrightarrow & aZ_n + c \equiv \left[ a^{n+1} Z_0 + c \frac{a^{n+1} - a}{a - 1} + c \right] \text{mod } m \\
\Leftrightarrow & aZ_n + c \equiv \left[ a^{n+1} Z_0 + c \frac{a^{n+1} - 1}{a - 1} \right] \text{mod } m \quad (2)
\end{aligned}$$

Dado que la relación de congruencia es, en particular, una relación de equivalencia se tiene la transitividad y por las ecuaciones (1) y (2) concluimos la demostración al obtener:

$$Z_{n+1} \equiv \left[ a^{n+1} Z_0 + c \frac{a^{n+1} - 1}{a - 1} \right] \text{mod } m$$

■

2. ¿Qué se puede decir de el periodo de  $Z_i \equiv aZ_{i-1} \bmod m$  con  $a = 630,360,016$  y  $m = 2^{31} - 1$

Solución:

Dado que es un GLC multiplicativo no cumple el teorema del periodo completo ( $c = 0$  por lo que no es primo relativo con  $m$ ) de forma que el periodo máximo que podría alcanzar es  $m-1$

3. Sin calcular ninguna  $Z_i$ , determinar cuál de los siguientes GLC's mixtos tienen periodo completo.

- (a)  $Z_i \equiv [13Z_i + 13] \bmod 16$
- (b)  $Z_i \equiv [12Z_i + 13] \bmod 16$
- (c)  $Z_i \equiv [13Z_i + 12] \bmod 16$
- (d)  $Z_i \equiv [Z_i + 12] \bmod 16$
- (e)  $Z_i \equiv [aZ_i + c] \bmod m$  con  $a = 2814749767109$ ,  $c = 59482661568307$   
y  $m = 2^{48}$

Solución:

Para resolver dicho problema se realizó una función en python que permite saber si un GLC tiene periodo completo o no, lo hace tras verificar que cumpla las 3 hipótesis del teorema del periodo completo, es decir, verifica:

- a) Que  $c$  y  $m$  son primos relativos
- b) Que si  $q$  es un número primo que divide a  $m$ , entonces  $q$  también divide a  $a - 1$  ( $a \equiv 1 \bmod q$  para cada  $q$  factor primo de  $m$ .)
- c) Finalmente, que si 4 divide a  $m$ , entonces 4 divide a  $a - 1$ . ( $a \equiv 1 \bmod 4$  si 4 divide a  $m$ ).

El programa está escrito en python 3 y el código fuente se muestra a continuación:

Tras ejecutar el programa en los ejercicios proporcionados se obtuvo que los generadores dados por las expresiones a), d) y e) tienen periodo completo mientras que los dados por b) y c) no, a continuación se muestran los resultados

```
1 a,c,m = 13,13,16
2 complete_period(a,c,m)
```

True

```
1 a,c,m = 12,13,16
2 complete_period(a,c,m)
```

Falla condición 2:

2 es primo y divide a  $m=16$  pero no a  $(a-1)=11$

Falla condición 3:

4 divide a  $m=16$  pero no a  $(a-1)=11$

False

```
1 a,c,m = 13,12,16
2 complete_period(a,c,m)
```

Los números no son primos relativos, su  $\text{MCD}(12,16) = 4$

False

```
1 a,c,m = 1,12,13
2 complete_period(a,c,m)
```

True

```
1 a,c,m = 2814749767109, 59482661568307, 2**48
2 complete_period(a,c,m)
```

True

4. Mostrar que el promedio de las  $U_i$ 's tomadas de un ciclo completo de un GLC de periodo completo es  $\frac{1}{2} - \frac{1}{m}$

Demostración:

Notemos que dado un generador de ciclo completo, si  $Z_i \equiv \left[ a^i Z_0 + c \frac{a^i - 1}{a - 1} \right] \bmod m$  entonces  $\{Z_i \mid 0 \leq i < m, \} = \{0, 1, \dots, m - 1\}$ .

Para probar dicha afirmación basta notar que por un lado  $\{Z_i \mid 0 \leq i < m, \} \subset \{0, 1, \dots, m - 1\}$  por la definición de los  $Z_i$ 's. Por otro lado  $\{0, 1, \dots, m - 1\} \subset \{Z_i \mid 0 \leq i < m, \}$  pues en caso contrario el generador no sería completo.

Con dicha observación, tenemos:

$$\begin{aligned}
 \frac{1}{m} \sum_{i=1}^m U_i &= \frac{1}{m} \sum_{i=1}^m \frac{Z_i}{m} \\
 &= \frac{1}{m^2} \sum_{i=1}^m Z_i \\
 &= \frac{1}{m^2} \sum_{i \in \mathbb{N}, i < m} i \\
 &= \frac{1}{m^2} \frac{(m-1)(m)}{2} \\
 &= \frac{(m-1)}{2m} \\
 &= \frac{m}{2} - \frac{1}{2m}
 \end{aligned}$$

■

5. Generar 10,000 números con  $U(0, 1)$  de Excel. Hacer un breve estudio para probar la calidad de los generadores; aplicar las pruebas de uniformidad e independencia a cada conjunto de datos. Resumir resultados en NO MAS de 2 cuartillas, incluyendo gráficas. De acuerdo a tus resultados, ¿cómo calificarías al generador de Excel?

6. Probar que la parte fraccional de la suma de uniformes en  $[0, 1]$ :  $U_1 + U_2 + \dots + U_k$  es también uniforme en el intervalo  $[0, 1]$ .

Demostración:

Comencemos por notar que, dadas  $U_1$  y  $U_2$  variables aleatorias con distribución uniforme, entonces la parte fraccional de  $U_1 + U_2$  tiene dicha distribución.

Denotemos por  $\{x\} = x - \lfloor x \rfloor$  la parte fraccional de  $x$ , sabemos que la densidad de  $U = U_1 + U_2$  es la siguiente:

$$f_U(x) = \begin{cases} x & 0 \leq x \leq 1 \\ 2 - x & 1 < x \leq 2 \end{cases}$$

Asimismo, la distribución de  $\{U\} = \{U_1 + U_2\}$  está dada por:

$$\begin{aligned} F_{\{U\}}(u) &= P\{\{U\} \leq u\} \\ &= P\{U - \lfloor U \rfloor \leq u\} \\ &= P\{U \leq u, 0 \leq U \leq 1\} + P\{U - \lfloor U \rfloor \leq u, 1 < U \leq 2\} \\ &= \int_0^u f_U(x) dx + \int_1^{1+u} f_U(x) dx \\ &= \int_0^u x dx + \int_1^{1+u} (2 - x) dx \\ &= \frac{u^2}{2} + 2u - \frac{(1+u)^2}{2} + \frac{1}{2} \\ &= u \end{aligned}$$

De esta manera la parte fraccional de  $U_1 + U_2$  tiene distribución uniforme en el intervalo  $(0, 1)$ .

Para la prueba del caso general  $U_1 + \dots + U_k$  se debe proceder por inducción; si se supone que la parte fraccional de la suma de  $n - 1$  variables aleatorias con distribución uniforme sigue dicha distribución, es suficiente notar que  $\{U_1 + U_2 + \dots + U_k\} = \{\{U_1\} + \{U_2 + \dots + U_k\}\}$  para concluir con la demostración. ■



7. Un generador de Fibonacci obtiene  $X_{n+1}$  a partir de  $X_n$  y  $X_{n-1}$  de la siguiente forma:

$$X_{i+1} = (X_i + X_{i-1}) \bmod m$$

Con  $X_0$  y  $X_1$  dados. Para  $m = 5$  solo dos ciclos son posibles, encontrarlos y al periodo.

Solucion:

Para la solución a dicho problema se implementaron las siguientes funciones en python3

La función `fibo` recibe como parámetros  $X_0$  y  $X_1$ , las raíces y  $m$ , el módulo. Y genera el los números producidos por la iteración hasta caer en un ciclo, como ejemplos:

```
1 fibo(1, 2, m=5)
'1,2,3,0,3,3,1,4,0,4,4,3,2,0,2,2,4,1,0,1'

1 fibo(3, 1, m=5)
'3,1,4,0,4,4,3,2,0,2,2,4,1,0,1,1,2,3,0,3'
```

Haciendo uso de dicha función, la siguiente función obtiene todos los posibles ciclos de el generador de fibonacci para un  $n$  dado, para  $n = 5$  tenemos los siguientes resultados:

```
1 get_different_cycles_fibonacci()
[{'Cicle': '0', 'X0': 0, 'X1': 0},
 {'Cicle': '0,1,1,2,3,0,3,3,1,4,0,4,4,3,2,0,2,2,4,1', 'X0': 0, 'X1': 1},
 {'Cicle': '1,3,4,2', 'X0': 1, 'X1': 3}]
```

Notamos que, además del ciclo trivial, hay 2 ciclos distintos.

8. Genera 10,000 números con una semilla de  $Z_0 = 1$  usando el generador  $Z_n = 75Z_{n-1} \bmod (2^{31} - 1)$  Clasifica los números en 10 celdas de igual tamaño y prueben por uniformidad usando la prueba  $\chi^2$  con un nivel de confianza del 90%. Aplicar también la prueba de rachas.

Solución

```
library(tidyverse)
library(randtests)

## Error in library(randtests): there is no package called 'randtests'
```

```
GLC = function(z0,a,c,m,k){
  Ui = rep(0,k)
  Ui[1] = z0/m
  z = (z0*a + c)%%m
  for (i in 2:k){
    Ui[i] = z/m
    z = (z*a + c)%%m
  }
  return (as.data.frame(Ui))
}

df = GLC(1,7^5,0,2^31-1,10000)
unif = df$Ui

h = hist(unif, breaks = 10, right = FALSE, plot = FALSE)
breaks_cdf <- punif(h$breaks)
null.probs <- breaks_cdf[-1] - breaks_cdf[-length(breaks_cdf)]
print(chisq.test(h$counts, p = null.probs, rescale.p = T))

##
## Chi-squared test for given probabilities
##
## data: h$counts
## X-squared = 6.676, df = 9, p-value = 0.6708
```

Concluimos con un nivel de significancia del 90 %, dado que el valor  $p$  es mayor que .1, que los números generados siguen una distribución uniforme.

Ahora apliquemos la prueba de rachas, por default la función `runs.test` del paquete *randtests* de R realiza la prueba de rachas de *Wald-Wolfowitz*.

```
runs.test(unif, plot = T)

## Error in runs.test(unif, plot = T): konnte Funktion "runs.test"
nicht finden
```

Notamos que no hay argumentos para rechazar la hipótesis alternativa de no aleatoriedad puesto que el  $p$ -value es mayor que .1.

9. Aplicar a los datos del ejercicio las pruebas de correlación, gaps y poker.  
Comencemos por aplicar la prueba de poker, por default las *manos* son de tamaño 5.

```
library(randtoolbox)

## Loading required package: rngWELL
## This is randtoolbox. For overview, type 'help("randtoolbox")'.

poker.test(unif)

##
##   Poker test
##
## chisq stat = 1.4, df = 4, p-value = 0.85
##
##   (sample size : 10000)
##
## observed number  3 204 956 755 82
## expected number  3.2 192 960 768 77
```

La prueba de poker no presenta argumento para rechazar la hipótesis de no aleatoriedad puesto que se reporta un  $p\text{-value} = .85$ .

Ahora realicemos la prueba de gaps. Por default, la función *gap.test* de R considera como gap al intervalo  $[0,.5]$ , realizando la prueba obtenemos:

```
gap.test(unif)

##
##   Gap test
##
## chisq stat = 7.6, df = 13, p-value = 0.87
##
##   (sample size : 10000)
```

```
##
## length observed freq theoretical freq
## 1 1207 1250
## 2 642 625
## 3 313 312
## 4 161 156
## 5 70 78
## 6 34 39
## 7 22 20
## 8 12 9.8
## 9 5 4.9
## 10 1 2.4
## 11 1 1.2
## 12 0 0.61
## 13 1 0.31
## 14 0 0.15
```

Observamos que el p-value es igual a .87 de forma que no tenemos argumentos para rechazar la hipótesis de no aleatoriedad. Sin embargo, realizaremos la prueba de gaps una vez más pero ahora modificando el intervalo a  $[.3, .6]$

```
gap.test(unif, lower = .3, upper = .6)

##
## Gap test
##
## chisq stat = 15, df = 7, p-value = 0.037
##
## (sample size : 10000)
##
## length observed freq theoretical freq
## 1 1498 1470
## 2 418 441
## 3 138 132
## 4 35 40
## 5 17 12
```

```
## 6    1    3.6
## 7    4    1.1
## 8    0    0.32
```

Observamos que el p-value es igual a 0.037 de forma que rechazamos la hipótesis de aleatoriedad.

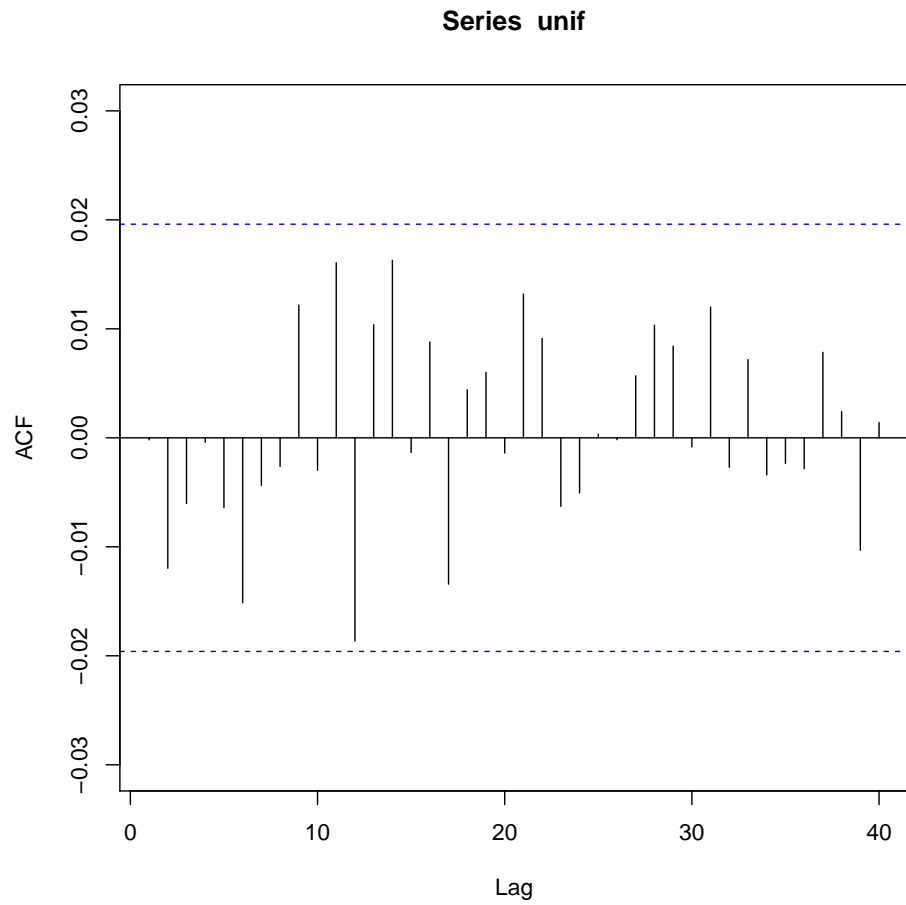
Finalmente, realizaremos la prueba de autocorrelación. Para ello se obtendrá la función de autocorrelación utilizando el paquete *forecast*.

```
library(hwwntest)

## Error in library(hwwntest): there is no package called 'hwwntest'

library(forecast)

#Obteniendo la función de autocorrelación
ACF = Acf(unif)
```



```
#Realizando el test de Bartlett
rho <- ACF$acf[2]
bt <- sqrt(length(unif))*rho
p.value = 1-pnorm(bt)/2
print(p.value)

## [1] 0.75

#Test de Bartlett para ruido blanco
bartlettB.test(unif)
```

```
## Error in bartlettB.test(unif): konnte Funktion "bartlettB.test"
nicht finden

#Test de Box-Pierce
Box.test(unif,lag=5,type="Box-Pierce")

##
## Box-Pierce test
##
## data:  unif
## X-squared = 2, df = 5, p-value = 0.8

#Test de Ljung-Box
Box.test(unif,lag=5,type="Ljung-Box")

##
## Box-Ljung test
##
## data:  unif
## X-squared = 2, df = 5, p-value = 0.8
```

Como se muestra en la gráfica, la función de autocorrelación no presenta valores extremos, asimismo, ninguna prueba de autocorrelación presenta argumentos para rechazar la hipótesis de aleatoriedad.



10. Generar 1500 números del generador RANDU. Hacer una prueba de Kolmogorov-Smirnov al 95 % de confianza.

Solución:

A continuación se realizara la prueba de Kolmogov-Smirnof a los dígitos generados por el generador RANDU con dos semillas distintas, 100 y 32.

```
randu1 = GLC(100,2^16 +3,0, 2^31,1500)
print(ks.test(randu1$Ui,"punif"))

##
## One-sample Kolmogorov-Smirnov test
##
## data: randu1$Ui
## D = 0.01, p-value = 0.9
## alternative hypothesis: two-sided

randu2 = GLC(32,2^16 +3,0, 2^31,1500)
print( ks.test(randu2$Ui,"punif") )

##
## One-sample Kolmogorov-Smirnov test
##
## data: randu2$Ui
## D = 0.04, p-value = 0.01
## alternative hypothesis: two-sided
```

Los resultados arrojados por la prueba a un nivel del 95 % de confianza no presentan argumentos para rechazar la hipótesis nula (de uniformidad) en el primer caso (semilla  $X_0 = 100$ ) puesto que el valor p es igual a .9351.

Asimismo, en el segundo caso ( $X_0 = 32$ ) se rechaza la hipótesis nula puesto que el valor p es .01358.

11. La página The number e to one million digits contiene el primer millón de dígitos de e (pueden usar cualquier otra página). Considerando estos dígitos:

- Realizar un histograma y verificar la hipótesis de que los dígitos corresponden a una distribución uniforme discreta.
- Verificar independencia de los dígitos, considerando las pruebas de gaps, de poker y de rachas. Una idea de ver los datos está en la siguiente imagen (esta está hecha para  $\pi$ ):

Comenzamos con la lectura de los dígitos:

```
decimals_e = readChar('e.txt', file.info('e.txt')$size)

## Warning in file(con, "rb"): kann Datei 'e.txt' nicht öffnen:
## No such file or directory
## Error in file(con, "rb"): kann Verbindung nicht öffnen

decimals_e = gsub('\n', '', decimals_e)

## Error in gsub("\n", "", decimals_e): Objekt 'decimals_e'
## nicht gefunden

df = data.frame("val" = unlist((strsplit(decimals_e, ''))))

## Error in strsplit(decimals_e, ""): Objekt 'decimals_e' nicht
## gefunden

df$val = as.numeric(as.character(df$val))

## Error in '$<-.data.frame'('*tmp*', val, value = numeric(0)):
## replacement has 0 rows, data has 10000
```

Realizamos un histograma de los mismos:

```
ggplot(df, aes(val)) +  
  geom_histogram(breaks = seq(-.5,9.5,by = 1)) +  
  labs(title="Histograma de primer millón de dígitos de e",  
        y="Frecuencia", x="Dígito")  
  
## Error in FUN(X[[i]], ...): Objekt 'val' nicht gefunden
```

Realizamos una serie de gráficas qq-plot

```
graf.teorica <- function(fun.quan,x,tit,...){  
  z <- sort(x,decreasing=F)
```

```

    plot(fun.quan(ppoints(z),0,10),z,main=tit,xlab =
        "dist. teorica",ylab = "datos")
    abline(a=0,b=1)
}

par(mfrow = c(2,2))

graf.teorica(qunif, head(df$val,20), tit = "20 dígitos")

## Warning in is.na(x): is.na() auf nicht-(Liste oder Vektor)
## des Typs 'NULL' angewendet
## Error in if (n > 0) (1L:n - a)/(n + 1 - 2 * a) else numeric():
## Argument hat Länge 0

graf.teorica(qunif, head(df$val,50), tit = "50 dígitos")

## Warning in is.na(x): is.na() auf nicht-(Liste oder Vektor)
## des Typs 'NULL' angewendet
## Error in if (n > 0) (1L:n - a)/(n + 1 - 2 * a) else numeric():
## Argument hat Länge 0

graf.teorica(qunif, head(df$val,125), tit = "125 dígitos")

## Warning in is.na(x): is.na() auf nicht-(Liste oder Vektor)
## des Typs 'NULL' angewendet
## Error in if (n > 0) (1L:n - a)/(n + 1 - 2 * a) else numeric():
## Argument hat Länge 0

graf.teorica(qunif, head(df$val,10000), tit = "10000 dígitos")

## Warning in is.na(x): is.na() auf nicht-(Liste oder Vektor)
## des Typs 'NULL' angewendet
## Error in if (n > 0) (1L:n - a)/(n + 1 - 2 * a) else numeric():
## Argument hat Länge 0

```

El test de  $\chi^2$  a un nivel de 95 % de confianza:

```
punifdisc <- function(q, min=0, max=9) ifelse(q<min,
0, ifelse(q>=max, 1, (floor(q)-min+1)/(max-min+1)))
qunifdisc <- function(p, min=0, max=9) floor(p*(max-min+1))

h1 <- hist(df$val, breaks = seq(-.5,9.5,by = 1), plot = F)

## Error in hist.default(df$val, breaks = seq(-0.5, 9.5, by
= 1), plot = F): 'x' must be numeric

breaks_cdf <- punifdisc(h1$breaks)

## Error in ifelse(q < min, 0, ifelse(q >= max, 1, (floor(q)
- min + 1)/(max - : Objekt 'h1' nicht gefunden

null.probs <- breaks_cdf[-1] - breaks_cdf[-length(breaks_cdf)]
a <- chisq.test(h1$counts, p = null.probs, rescale.p = T)

## Error in is.data.frame(x): Objekt 'h1' nicht gefunden

a

## Error in eval(expr, envir, enclos): Objekt 'a' nicht gefunden
```

Dado que el valor p es mayor que .05 no existen argumentos para rechazar la hipótesis de uniformidad. Continuemos con las pruebas de independencia.

```
runs.test(unif)

## Error in runs.test(unif): konnte Funktion "runs.test" nicht
finden

#poker.test(unif)
#gap.test(unif)
```