Guía 1: Introducción a R/RStudio y simulaciones

Toma de decisiones 2020

Guillermo Solovey

Nota

Algunos ejercicios de esta guía se pueden resolver con "lápiz y papel" sabiendo un poco de probabilidad. Pero no es eso lo que nos importa ahora. La idea es resolverlos haciendo simulaciones numéricas, por al menos dos razones: es una buena práctica para aprender a programar, se pueden comparar las simulaciones con el resultado exacto (si existe).

En algunos ejercicios, tienen partes del código que los resuelve, con algunas líneas para completar. Pueden usar eso como ayuda, copiando y pegando el código en el editor de RStudio.

Hay muchas maneras de resolver los ejercicios correctamente. Si quieren probar algo diferente a lo propuesto, adelante. Después comparamos y discutimos las diferencias.

Ejercicio 1

Escribir una función que simule N tiradas de un dado y devuelva la suma de los valores obtenidos. Repetir el proceso 10000 veces y hacer un histograma con los resultados obtenidos. Ayuda: cada tirada del dado corresponde a elegir uno de los siguientes números con igual probabilidad: {1, 2, 3, 4, 5, 6}.

A- Usando la función sample() de R simule una tirada de un dado.

```
sample(___, ___, replace = ___)
```

B- Crear una función que realice la suma de los resultados de N dados. Evaluar la función en distintos valores de N.

```
suma_dados <- function(N){
  todos     <- sample(c(1,2,3,4,5,6), ___, replace = ___)
  respuesta <- ___
  return(___)
}

suma_dados(___)
```

C- Escribir un loop que repita el proceso anterior 10000 veces, guardando el resultado de cada 'experimento' en un vector.

```
Nrep   <- 10000
output <- rep(NA, Nrep)  # Inicializo el vector que va a guardar los datos

# Repito el proceso suma_dados 10000 veces
for(i in 1:Nrep){
  output[___] <- suma_dados(___)
}
```

D- Haga un histograma de los resultados anteriores (investigue cómo usar la función hist() de R).

Ejercicio 2

En un grupo de N personas, ¿cuál es la probabilidad de que al menos dos personas cumplan años el mismo día?

A- Simular las fechas de cumpleaños de N=50 personas y guardarlas en un vector.

```
N <- 50
cumples <- sample(___, N, replace = ___)
```

B- Evaluar si hubo o no una coincidencia. ¿Qué hace la función 'unique()'?

```
length(unique(cumples)) < N
```

C- Repetir el proceso anterior 10000 veces y contar todos los casos en los que hubo al menos 1 coincidencia

```
# variable que va a contar las coincidencias
n_coincidencias <- 0

# simula 10000 grupos de N=50 personas y verifica si hubo coincidencias o no
Nrep <- 10000
for(i in 1:Nrep){
  cumples <- sample(seq(1,365,1), N, replace = TRUE)
  if(length(unique(cumples)) < N){
    n_coincidencias <- ___
  }
}

# calcula la probabilidad estimada de coincidencias y la imprime en la consola
p_coincidencias <- n_coincidencias / Nrep
print(p_coincidencias)
```

D- Crear una función 'pcumples' que tenga dos inputs: el número de personas (N) y el número de repeticiones del experimento (Nrep). La salida de la función debe ser la probabilidad de tener al menos una coincidencia.

```
pcumples <- function(N=50, Nrep=10000){
  # 50 y 10000 son los valores por defecto de la función
  ___
  ___
  return(___)
}
```

E- La función pbirthday() de R calcula la probabilidad de coincidencia en un grupo de N personas. Comparar el resultado anterior con el que se obtiene con la función pcumples que crearon en el ítem anterior.

F- Hacer un gráfico que muestre la probabilidad de coincidencia en función del número de personas en el grupo. Puede usar la función pbirthday() o pcumples().

```
# Define el vector con los tamaños de los grupos.
N_vec <- ___

# ejecuta la función para cada elemento del vector anterior
for (i in 1:length(Nvec)){
  p_c[___] <- pbirthday(___)
}
```

```
# una forma equivalente de hacer el loop anterior es usando la función sapply
p_c[___] <- sapply(___)

# crea el gráfico
plot(___ ~ ___)
```

Ejercicio 3

Si en una votación entre dos candidatos, n votantes votan por A y m votantes votan por B (con n>m) la probabilidad de A le vaya ganando a b a lo largo de todo el escrutinio es $(n-m)/(n+m)$. Nota: el que me presentó este problema es Pablo Groisman en Twitter. Pueden ver la demostración acá. En este ejercicio tienen que hacer simulaciones de escrutinios, evaluar si A le gana a B a lo largo de todo el escrutinio y comparar el resultado obtenido con el cálculo exacto.

A- Crear un vector que contenga todos los votos, usando la notacion +1 para votos para el candidato A y -1 para el candidato B (¿por qué es conveniente esto?)

```
n <- 500 # numero de votos para A
m <- 400 # numero de votos para B
votos <- c( ___, ___ )
```

B- Un escrutinio es un posible orden en el que van apareciendo las boletas. Crear un escrutinio y evaluar si A se mantuvo siempre al frente.

```
# realizar un escrutinio y evaluar si A se mantuvo al frente durante todo el escrutinio.
# (puede servir usar la funcion cumsum. Que hace?)
escrutinio <- sample(___, n+m, replace = FALSE)
resultado  <- cumsum(___)
# evaluar si A se mantuvo siempre al frente
A_gana_siempre <- ___ # TRUE si A gana siempre, FALSE si no.
```

C- Repetir el proceso anterior 1000 veces y contar en cuantos escrutinios A se mantuvo al frente en todo el escrutinio

```
cuenta <- 0
Nrep   <- 1000
for (i in 1:Nrep){

  escrutinio <- sample(votos, n+m, replace = FALSE)
  resultado  <- cumsum(escrutinio)
  if ( ___ ){
    ___
  }

}

p <- ___
print(p)
print((n-m)/(n+m))
```

D- Suponga ahora que 505 personas votaron por A y 495 personas por B. ¿Cuál es la probabilidad de que en el escrutinio vaya ganando B desde el comienzo hasta que se contaron 800 votos? Compare con el caso en que va ganando A, también hasta contar 800 votos.

```
n <- 505 # numero de votos para A
m <- 495 # numero de votos para B
```

```
votos <- c( rep(1,n), rep(-1,m) )

# cuentaA va a contar cuántas veces ocurre que A va ganando
# hasta contar 800 votos
cuentaA <- 0
cuentaB <- 0  # esta hace lo mismo para B
Nrep   <- 10000
for (i in 1:Nrep){

  escrutinio <- sample(votos, n+m, replace = FALSE)
  resultado  <- cumsum(escrutinio)
  if ( ___ ){
    cuentaA <- cuentaA + 1
  }
  if ( ___ ){
    cuentaB <- cuentaB + 1
  }

}

pA <- cuentaA / Nrep
pB <- cuentaB / Nrep
print(pA)
print(pB)
```

E- En el 2017, en la provincia de Buenos Aires, la elección de senadores se definió por una diferencia pequeña entre los dos candidatos más votados. Sin embargo, durante el escrutinio provisorio, uno de ellos iba siempre al frente. ¿Qué suposición hicimos en las simulaciones que podría no ser cierta en un escrutinio real? [1]

Ejercicio 4

Suele decirse que una tostada con manteca que se cae de la mesa siempre impacta el suelo del lado de la manteca. Para probarlo, alguien repite el experimento 30 veces, tomando todos los recaudos para que las condiciones de cada experimento sean iguales. El resultado fue que, de las 30 veces que tiró la tostada, 22 veces la tostada cayó del lado de la manteca. ¿Qué podemos concluir?

A- Si la manteca untada no tuviera ninguna relevancia en la caída de la tostata (o sea, si la probabilidad de que caiga del lado de la manteca fuese 0.5) ¿qué probabilidad habría de encontrar 22 veces (sobre 30) la tostada cayó del lado de la manteca? ¿Y la probabilidad de encontrar que **al menos** 22 veces ocurre eso?

B- Si la probabilidad de caer del lado de la manteca fuese 0.7, ¿qué probabilidad habría de encontrar que 22 veces (sobre 30) la tostada cayó del lado de la manteca?

Ejercicio 5

Regla de la mayoría. Un modelo simple de propagación de opiniones supone que hay N personas que mantienen dos estados posibles de opinión: {+1, -1}. Inicialmente el número de personas en el estado +1 es n1<N. Luego, en pasos sucesivos, 3 personas seleccionadas al azar interactúan de forma tal que adoptan la opinión mayoritaria entre los 3. Por ejemplo, si hay dos personas con opinión +1 y una con opinión -1, luego de la interacción, la opinión de los tres será +1.

[1] Si quieren saber qué pasó ese día: Antenucci, P., Mascioto, J. M., & Page, M. (2017). PASO 2017 en la provincia de Buenos Aires: el recuento provisorio explicado. Revista SAAP. Publicación de Ciencia Política de la Sociedad Argentina de Análisis Político, 11(2), 341-364.

A- Crear el estado inicial de cada persona, usando un vector 'Op'

```
N  <- 10   # numero total de personas
n1 <- 4    # numero de personas que inicialmente opinan +1
Op <- c( rep(1, n1), ___)
```

B- Hacer una ronda de interacción. Esto es, elegir tres personas al azar, buscar la opinión mayoritaria y luego actualizar la opinión de cada uno.

```
n_int     <- sample(1:N, 3, replace = FALSE)
Op[n_int] <- ifelse( ___, 1, -1 )
```

C- Repetir el proceso anterior hasta que se llegue a un consenso.

```
Op       <- ___
consenso <- 0
while( consenso != 1 ){

  n_int     <- sample(1:N, 3, replace = FALSE)
  Op[n_int] <- ifelse( ___ )

  if (___ == 1){
    consenso  <- 1
    Oconsenso <- ___
  }

}
```

D- Crear una función que dado el número de personas (N), la fracción inicial que opina +1 (p), calcule la probabilidad de que el consenso se establezca en que todos opinen +1 (**P1**) realizando 100 experimentos simulados.

```
p_consenso <- function(N, n1){

  output   <- 0

  for (i in 1:100){
    ---
    ---
    ---

    if (Oconsenso == 1){
      output <- output + 1
    }

  }

  return(output/100)
}
```

E- Graficar **P1** en función de p para los siguientes valores de n1 = {1, 2, 3, . . . , 10}.

```
x <- 1:N
y <- rep(NA, length(x))

for (i in 1:length(x)){
  y[i] <- p_consenso(10, x[i])
}
```

```r
plot(x, y)
```

Ejercicio 6

Encontrar la recta $y = a + bx$ que mejor ajusta a los siguientes puntos (x_i, y_i) en el sentido de minimizar la distancia cuadrática. Para eso la idea es probar con un conjunto de valores de a y un conjunto de valores de b y para cada par de valores (a, b) calcular el error.

A- Hacer un gráfico de los datos siguientes

```r
x <- c(-5.0, -3.9, -2.8, -1.7, -0.6, 0.6, 1.7, 2.8, 3.9, 5.0)
y <- c(9.1, 5.5, 11.2, 6.7, 1.3, 5.7, 1.0, -4.4, -4.0, -11.0)
```

B- Armar la tabla de posibles valores de a y b. ¿Qué hace la función 'expand.grid' y qué rango de valores de a y b conviene usar?

```r
p <- expand.grid(a = ___, b = ___)
```

C- Calcular el error cuadrático para cada par de valores (a, b)

```r
# calcula el error cuadratico medio para cada par de valores a y b
error2 <- rep(NA, nrow(p))
for (k in 1:nrow(p)){

  a         <- p$a[k]
  b         <- p$b[k]

  ___
  for (i in 1:length(x)){
      error2[k] <- ___
  }
}
```

D- Encontrar los valores (a, b) que minimizan el error cuadrático y con esos valores graficar la recta correspondiente por sobre los puntos graficados antes.

```r
kmin   <- which.min(error2)
a.best <- ___
b.best <- ___

abline(a.best, b.best)
print(a.best)
print(b.best)
```

E- Comparar el resultado obtenido con la función de R 'lm' que calcula analíticamente los mejores parámtros.

```r
lm(y~x)
```

Ejercicio 7

Problema de Monty Hall. Un participante de un concurso tiene que elegir entre tres puertas. Detrás de una de ellas hay un premio. Al elegir una puerta, el conductor del concurso le señala cuál de las otras dos puertas seguro *no* tiene el premio. El participante tiene la opción de quedarse con su opción inicial o cambiar a la otra puerta. ¿Qué le conviene? Calcular la probabilidad de éxito, comparando simulaciones en las que el participante elige quedarse y otras en las que decide cambiar.