Segundo Laboratorio - Introducción al R

Bioestadística 2020

Bucles

Los bucles son una herramienta muy usada en cualquier lenguaje de programación. Si necesitamos ejecutar muchas órdenes (hasta varios millones de ellas) similares, no podemos escribirlas todas. Afortunadamente, le podemos brindar a R un conjunto de índices, y una orden para cada índice; R ejecutará sucesivamente todas esas órdenes. Tomemos como ejemplo sumar todos los números del 1 al 1 millón (1x10^6). El enunciado que usaremos es el *for(){}*: dentro del paréntesis, entramos un conjunto de índices, el cual R recorrerá en orden; dentro de los corchetes, entramos las órdenes que se ejecutarán para cada índice.

A continuación veremos el código para sumar de todos los números del 1 al 1 millón. Haremos varios comentarios sobre el mismo.

```
suma = 0
# Esta es la variable que guardará el resultado final.
# Observar que la tenemos que 'inicializar' antes del enunciado 'for'.
for(i in 1:(1000000)){
  # La variable 'i' se actualiza para cada orden,
  # al igual que todas las variables creadas dentro del enunciado.
  # CUIDADO: si la variable 'i' ya existía, será sobreescrita al recorrer los índices.
  suma = suma + i
  # Observar que el valor de 'suma' se actualiza en cada orden:
  # el valor de 'suma' más 'i' es el nuevo valor de 'suma'.
  # Dicho de otra manera, en cada orden dentro del buclce, el valor de 'suma'
  # se sobreescribe por el mismo valor más el valor del índice 'i'.
}
suma # Muestra el resultado final.
```

```
## [1] 500000500000
```

```
i # Último índice usado.
```

```
## [1] 1000000
```

Para ver mejor lo que está sucediendo, sumaremos los números del 1 al 10 de la misma manera, e imprimiremos el valor de *suma* en cada paso. Usaremos la función *cat()* para imprimir texto y valores de variables en una misma linea.

```
suma = 0
v.ind = 1:10
for(i in v.ind){
  cat("Paso ", i, ":", "\n")
  # Usamos el comando '\n' para pasar a la siguiente linea.
  cat("'i' = ", i, "\n")
  # Aquí se apreciará que el valor de 'i' fue actualizado 10 veces,
  # y en qué orden se hizo.
  cat("'suma' antes de actualizar = ", suma, "\n")
  suma = suma + i
  cat("'suma' después de actualizar = ", suma, "\n", "\n")
}
```

```
## Paso  1 :
## 'i' =  1
```

```
## 'suma' antes de actualizar =  0
## 'suma' después de actualizar =  1
##
## Paso  2 :
## 'i' =  2
## 'suma' antes de actualizar =  1
## 'suma' después de actualizar =  3
##
## Paso  3 :
## 'i' =  3
## 'suma' antes de actualizar =  3
## 'suma' después de actualizar =  6
##
## Paso  4 :
## 'i' =  4
## 'suma' antes de actualizar =  6
## 'suma' después de actualizar =  10
##
## Paso  5 :
## 'i' =  5
## 'suma' antes de actualizar =  10
## 'suma' después de actualizar =  15
##
## Paso  6 :
## 'i' =  6
## 'suma' antes de actualizar =  15
## 'suma' después de actualizar =  21
##
## Paso  7 :
## 'i' =  7
## 'suma' antes de actualizar =  21
## 'suma' después de actualizar =  28
##
## Paso  8 :
## 'i' =  8
## 'suma' antes de actualizar =  28
## 'suma' después de actualizar =  36
##
## Paso  9 :
## 'i' =  9
## 'suma' antes de actualizar =  36
## 'suma' después de actualizar =  45
##
## Paso  10 :
## 'i' =  10
## 'suma' antes de actualizar =  45
## 'suma' después de actualizar =  55
##
```

```
suma
```

```
## [1] 55
```

El vector de índices puede contener cualquier cosa; incluso estar ordenado de cualquier manera. Es un buen ejercicio estudiar cómo funciona este bucle.

```r
v.nombres = c("Eustaquio", "Susana", "Aníbal", "Romina")
for(nombre in v.nombres){
  # Recorremos todos los nombres de 'v.nombres' en orden,
  # usando la variable interna 'nombre'.

  if(! nombre == v.nombres[1]){
    # Entra siempre que no sea el primer nombre.
    cat("-","¡",nombre,"! \n \n")
  }

  cat("-",nombre, "robó pan de la casa de Juan.", "\n")

  if(!nombre == v.nombres[length(v.nombres)]){
    # No entro en este paso si estoy en el último nonmbre de la lista.
    cat("-","Quién, ¿yo? \n")
    cat("-","Sí, tú. \n")
    cat("-","Yo no. \n")
    cat("-","¿Pues quién? \n")
  }

}
```

```
## - Eustaquio robó pan de la casa de Juan.
## - Quién, ¿yo?
## - Sí, tú.
## - Yo no.
## - ¿Pues quién?
## - ¡ Susana !
##
## - Susana robó pan de la casa de Juan.
## - Quién, ¿yo?
## - Sí, tú.
## - Yo no.
## - ¿Pues quién?
## - ¡ Aníbal !
##
## - Aníbal robó pan de la casa de Juan.
## - Quién, ¿yo?
## - Sí, tú.
## - Yo no.
## - ¿Pues quién?
## - ¡ Romina !
##
## - Romina robó pan de la casa de Juan.
```

Ejercicio 1

Escribir un código que retorne el producto de los números naturales entre el 1 y el 50. Usar la función *prod* para obtener el mismo producto, y comparar.

Ejercicio 2

Escribir un código que retorne el promedio de *sen(i)*, para los i entre 0 y 100. Hacer lo mismo usando la función *mean*.

Ejercicio 3

Modificar el ejercicio 3 para que haga el promedio de los *sen(i)*, para i entre 0 y 100, tal que *sen(i)>0*. Hacer lo mismo usando al función *mean*.

Ejercicio 4

Crear un vector v de largo 100 que cumpla *v[i]=iˆ2 - i + 1*.

Simulación de variables aleatorias

Muchas veces, necesitamos realizar cálculos estadísticos donde no hay teoría matemática que nos pueda ayudar. Por ejemplo, un teorema matemático puede ofrecernos una herramienta que funciona perfectamente, pero solo si las hipótesis se cumplen. De lo contrario, nosotros no podemos predeecir lo que sucederá. Un ejemplo más concreto: si tiramos una moneda 1000 veces, y todas las tiradas son independientes una de otra, sabemos que, aproximadamente 500 veces saldrá cara, y 500 veces saldrá número. Incluso, podemos saber el valor máximo y mínimo de la cantidad de caras que pueden salir (donde la probabilidad de que la cantidad de caras no cumpla dicho rango es tan chica como deseemos). Sin embargo, si no sabemos que las tiradas sean independientes, es posible que no haya un teorema que nos ayude a predecir qué proporción de caras y números aparecerán, ni el rango máximo y mínimo de la cantidad de caras. Para estos casos, lo que suele hacerse es simular el experimento muchas veces, y observar el comportamiento. A partir de esos resultados, podemos inferir probabilidades (casos favorables sobre casos posibles) y estadísticas. Por lo tanto, simular variables aleatorias es una herramienta fundamental. El software R nos permite simular muchos tipos de variables aleatorias. Veremos unos ejemplos que usaremos mucho.

Comenzaremos simulando variables aleatorias Bernoulli (sale 0 con probabilidad *1 - p*, sale 1 con probabilidad p). Este caso lo encontraremos como caso particular de simular una variable aleatoria Binomial (que simula n variables Bernoulli y suma los resultados).

```
?rbinom
```

```
## starting httpd help server ... done
```

```
r.ber = rbinom(n = 1, size = 1, prob = 1/2) #
# Fijando 'n = 1' se simula solo una V.A. Binomial.
# La variable 'size' indica cuántas V.A. Bernoulli sumaremos
# en cada una de las 'n' realizaciones.
# La variable 'prob' indica la probabilidad de obtener un 1
# en cada realización.
r.ber
```

```
## [1] 1
```

```
v.binom = rbinom(100, 5, 1/2)
v.binom
```

```
##   [1] 3 4 4 5 2 2 1 3 3 4 2 2 4 5 4 2 4 3 5 3 1 3 1 1 2 1 2 1 1 2 3 3 2 3 2
##  [36] 1 2 2 3 2 1 3 4 4 2 4 2 1 3 2 4 0 1 2 3 3 3 3 1 3 2 2 2 5 3 0 4 3 4 3
##  [71] 3 2 1 2 3 5 2 3 3 2 1 3 3 1 1 4 3 4 3 3 2 1 2 4 4 5 2 3 3 2
```

```
v.ber = rbinom(n = 1000, size = 1, prob = 1/2)
suma.ber = sum(v.ber)
# Cantidad de unos al simular una variable aleatoria
# Bernoulli 1000 veces de manera independiente.
# 'suma.ber' también puede ser interpretado como simular una
# variable aleatoria Binomial con parámetros '(n = 1000, p = 1/2)'.
suma.ber
```

```
## [1] 485
```

Si estamos interesados en elegir un número al azar del *1* al *n*, usamos la función *sample.int()*. Si tenemos un conjunto de valores ya seleccionado y queremos elegir uno al alzar, usamos la función *sample()*.

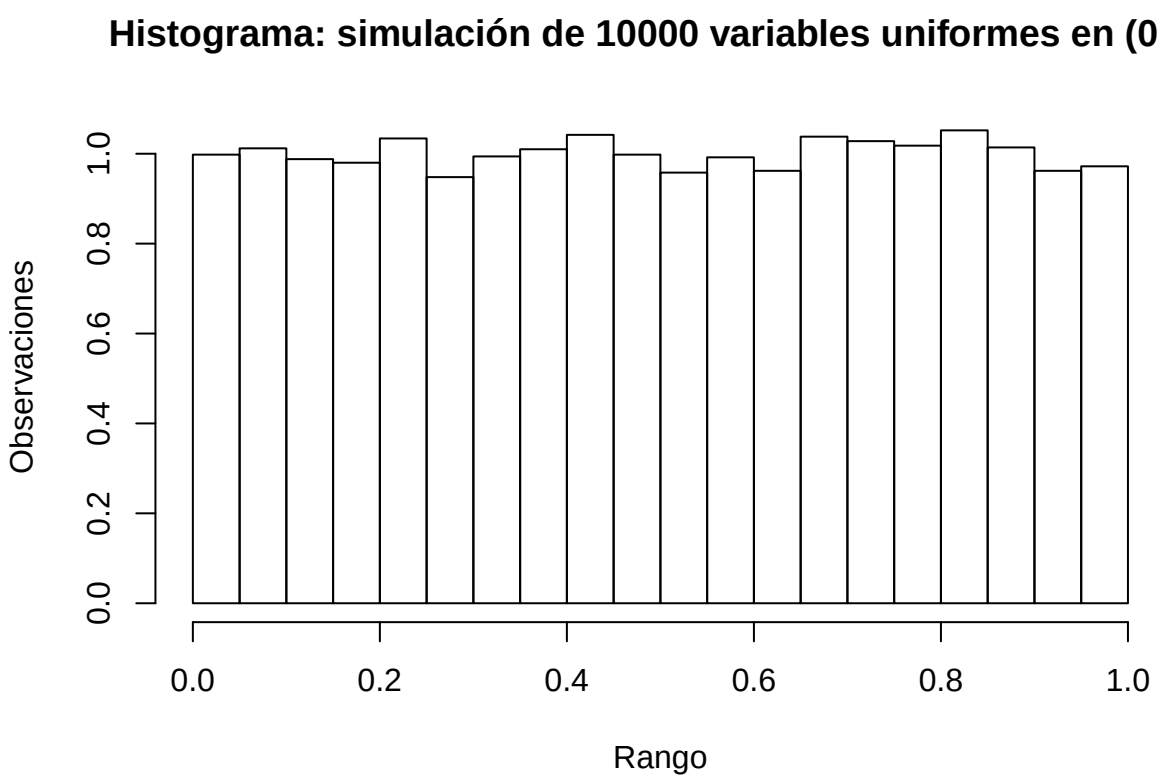
```
?sample
r.samp = sample.int(n = 10, size = 1)
# Número al azar entre 1 y 10.
# También podemos usar 'sample.int()' para obtener una permutación al azar.
r.perm = sample.int(n = 10, size = 10, replace = FALSE)
# Devuelve los números del 1 al 10 reordenados
# Podemos también cambiar la probabilidad de cada número de ser seleccionado
r.sel = sample.int(n = 10, size = 1, prob = c(1, 0*(1:9)))
# El vecotr de probabilidades es de largo 10, donde la primer entrada
# es igual a 1, y el resto es igual a cero.
# Por lo tanto, el resultado de 'r.sel' siempre será igual a 1.
# También podemos hacer sorteos con repetición
r.rep = sample(x = c("azul", "rojo"), size = 10, replace = TRUE)
```

Al simular variables aleatorias discretas, los dos casos anteriores son los más importantes, pero hay muchas opciones más.

```
?rpois # Simula variables aleatorias Poisson
?rgeom # Simula variables aleatorias Geométricas
?rhyper # Simula variables aleatorias Hipergeométricas
```

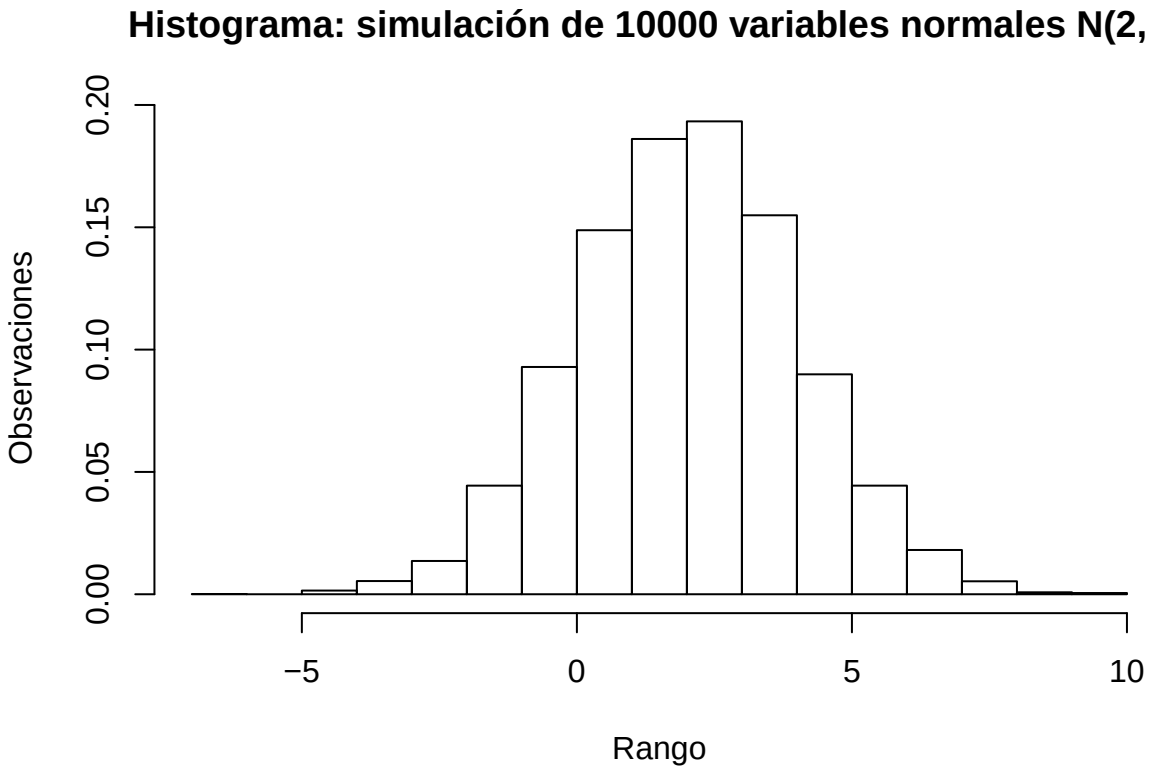
En el caso de necesitar simular una variable aleatoria continua, también tenemos muchas opciones. La función más importante es *runif()*, que nos permite elegir un número real al azar entre 0 y 1.

```
?runif
r.unif = runif(n = 1, min = 0, max = 1) # Número uniforme entre 0 y 1.
# Podemos elegir el rango de la variable aleatoria modificando los parámetros 'max' y 'min'.
r.nunif = runif(n = 10000, min = 0, max = 1)
hist(r.nunif, breaks = 20, freq = FALSE,
     main = 'Histograma: simulación de 10000 variables uniformes en (0, 1)',
     xlab = "Rango", ylab = "Observaciones")
```



Otra función importante es la que nos permite simular variables aleatorias normales.

```
?rnorm
r.norm = rnorm(n = 1, mean = 0, sd = 1)
# Aquí, 'n' representa la cantidad de simulaciones a realizar,
# 'mean' la media de las variables aleatorias, y 'sd' su desviación
# estándar (CUIDADO, no confundir con la varianza).
r.nnorm = rnorm(10000, mean = 2, sd = 2)
# Si la desviación estándar es 2, la varianza es 2^2 = 4.
hist(r.nnorm, breaks = 20, freq = FALSE,
     main = 'Histograma: simulación de 10000 variables normales N(2, 4)',
     xlab = "Rango", ylab = "Observaciones")
```



Otras opciones para simular variables aleatorias continuas:

```
?rexp # Simula variables aleatorias Exponenciales
?rchisq # Simula variables aleatorias Chi-Cuadrado
?rcauchy # Simula variables aleatorias Cauchy
?rt # Simula variables aleatorias T de Student.
```

Ejercicio 5

Consideremos 10000 realizaciónes de una variable aleatoria X normal $N(m,s)$, con $m=3$, y $s^2 = 4$. Calcular el promedio de las X que cumplan $-1<X<3$. ¿Coincide el resultado con lo esperado? Probar para distintos valores de m.

Ejercicio 6

Modificar el ejercicio 5 para que calcule el promedio de la variable aleatoria $Y = 2X - 1$ para el caso $m=0$ y $s^2 = 4$.

Ejercicio 7

Modificar el ejercicio 5 para calcular el promedio de las X que cumplan $-1<X<3$ en el caso $m=0$ y $s^2=4$.

Ejercicio 8

Realizar lo pedido en el ejercicio 5 100 veces, y guardar los resultados en un vector v.

Más sobre variables aleatorias

En la sección anterior vimos como simular variables aleatorias con los comandos *rnorn*, *runif*, *rbinom*. Ahora vamos a ver como obtener la distrinución para las distintas densidades. Para la normal

```
?pnorm
x=c(-1,-0.5,0,0.5,1)
p.norm = pnorm(x, mean = 0, sd = 1)
# Aquí, 'x' es un vector de valores para los que queremos calcular la distribución,
# 'mean' la media de las variables aleatorias, y 'sd' su desviación
# estándar (CUIDADO, no confundir con la varianza).
p.norm
```

```
## [1] 0.1586553 0.3085375 0.5000000 0.6914625 0.8413447
```

También podemos obtener los cuantiles para las distintas densidades. Para la normal

```
?qnorm
q=c(0.2,0.4,0.6,0.8)
q.norm=qnorm(q, mean = 0, sd = 1)
# Aquí, 'q' es un vector de valores para los que queremos calcular los cuantiles,
# 'mean' la media de las variables aleatorias, y 'sd' su desviación
# estándar (CUIDADO, no confundir con la varianza).
q.norm
```

```
## [1] -0.8416212 -0.2533471  0.2533471  0.8416212
```

Análogamente con *punif* y *qunif* podemos hacerlo para la uniforme, si usamos *pbinom* y *qbinom* lo obtenemos para la binomial.

Otro comando importante a tener en cuenta es el comando *summary*, que nos devuelve máximo, mínimo, promedio, mediana y cuantiles de un vector de datos.

```
?summary
x=rnorm(1000,3,2)
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -3.188   1.684   3.020   2.998   4.394   8.455
```

Ejercicio 9

Considerar n realizaciones de una variable aleatoria X normal *N(m,s)*, con *m=0*, y *s^2 = 1*. Calcular el porcentaje de realizaciones de X que están entre *qnorm(0.025)* y *qnorm(0.975)*. ¿Que pasa para este porcentaje cuando n es muy grande? (Considerar n=100,1000,10000,100000).

Ejercicio 10

Modificar el ejercicio anterior pero para otro porcentaje *alpha*. ¿Cuales serían los valores para calcularle el *qnorm*?.