Guía 5: Programación en R: funciones, iteraciones

Laboratorio de datos 2023 (comisión: G. Solovey)

1. Unir data frames

En esta parte de la guía se utilizarán datos ficticios generados aleatoriamente para emular parte de la estructura de datos que podría haber en un sistema de información de una institución educativa. En dicho sistema hay una tabla (dfEstudiantes) con ciertos datos de las/os estudiantes (legajo, edad, carrera) y otra tabla (dfNotas) con las notas de los exámenes de Biología y de Matemática (legajo, nota, materia).

El siguiente código permite crear las dos tablas que se usarán en esta guía:

```
library(dplyr)

set.seed(1010)
aux = rchisq(100, df=2)

dfEstudiantes = tibble(legajo = paste0("LE_",1:100),
                       edad = as.integer((80-18) * aux/max(aux) + 18),
                       carrera = sample(c("Ciencias Físicas",
                                          "Ciencias Matemáticas",
                                          "Ciencias de Datos",
                                          "Paleontología",
                                          "Ciencias Biológicas",
                                          "Ciencias de la Atmósfera",
                                          "Ciencias de la Computación",
                                          "Ciencias Geológicas",
                                          "Ciencias Químicas",
                                          "Ciencia y Tecnología de Alimentos",
                                          "Oceanografía"),
                                        100, replace = T))

dfNotas = rbind(tibble(legajo = sample(paste0("LE_",1:50),30),
                       nota = as.integer(runif(30,min=2,max=10)),
                       materia = "Biología"),
                tibble(legajo = sample(paste0("LE_",1:100),50),
                       nota = as.integer(runif(50,min=2,max=10)),
                       materia = "Matemática"))
```

1.1 Cuál es la nota promedio por carrera de la materia Biología?

1.2 Cuál es la cantidad de estudiantes por carrera que no rindió ninguno de los dos exámenes?

1.3 Cuál es la edad promedio de los estudiantes que rindieron al menos uno de los dos exámenes? Tener en cuenta que un/a mismo/a estudiante puede haber rendido más de un examen. Revisar el uso de la función `distinct()`.

1.4 Generar un gráfico que permita visualizar la distribución de notas por grupo etario y materia, considerando los grupos etarios: [18, 21], (21,27] y [27, 80).

2. Funciones

2.1 Escribir una función que calcule el rango de un vector numérico. Comparar con la función `range()` de R.

2.2 Escribir una función que calcule la media de un vector numérico. Comparar con la función `mean()` de R.

2.3 Escribir una función que calcule la proporción de valores faltantes (`NA`) en un vector numérico.

2.4 Modiicar la función de 2.3 para que reemplace los `NA` con un valor específico numérico (-1 por ejemplo).

2.5 Crear una función que reemplace valores especificados en un vector numérico (por ejemplo los -1) con NA.

2.6 Pensar qué operación realiza el siguiente script y transformarlo en una función. ¿Cuántos argumentos necesitaría? ¿Cómo la llamaría a la función?

```
round(x / sum(x, na.rm = TRUE) * 100, 1)
round(y / sum(y, na.rm = TRUE) * 100, 1)
round(z / sum(z, na.rm = TRUE) * 100, 1)
```

2.7 Escribir la función `ambos_na()` que toma dos vectores del mismo tamaño y devuelve los índices correspondientes a posiciones donde los dos vectores tienen `NA` . ¿Qué sucede con esta función si los vectores tienen distinto tamaño? Agregar una verificación y mensaje de texto que advierta que los vectores no tienen el mismo tamaño.

2.8 Escribir una función que devuelva un objeto gráfico de `ggplot2` que realice de forma incremental lo siguiente

- Dibuje un scatterplot de la variable ´x´ e ´y´ de un dataset.
- Agregar una recta correspondiente al mejor ajuste lineal (sin mostrar el sombreado de margen de error).
- Agregue un título

2.9 Pensar qué hace esta función y entendiendo su finalidad, sugerir un mejor nombre:

```
f2 <- function(lst, n) {
  length(lst) >= n
}
```

2.10 Si alguna de las funciones que escribiste en los ejercicios anteriores tenían nombres crípticos, cambiarlos por nombres más apropiados.

2.11 Escribir una función que filtre un data frame para seleccionar filas donde una columna específica contiene valores faltantes.

2.12 Escribir una función que calcule estadísticas resumidas (media, mediana, mínimo, máximo) para una columna numérica en un data frame.

2.13 Ídem el anterior pero agrupando por grupo (es decir, uno de los argumentos de la función debe ser la variable por la que hay que agrupar).

2.14 Escribir una función que genere un gráfico de dos variables numéricas de un data frame usando `geom_hex()` . Comprobar con un ejemplo.

3. Iteraciones

3.1 Utilice un `while` para encontrar la potencia menor de 2 que sea mayor que 1000.

3.2 Escribe un `while` para calcular la suma de todos los números naturales positivos menores que 100 que sean múltiplos de 7.

3.3 Crea un `while` para simular el crecimiento de una población. Comienza con 100 individuos y, cada año, la población se duplica. Calcula cuántos años se necesitan para que la población alcance los 1000.

3.4 Utilice un bucle mientras para encontrar el número primo más grande menor que 100.

4. Desafíos

4.1 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.

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(___, ___, 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.

D- Haga un histograma de los resultados anteriores.

4.2 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()`?

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
ggplot(____)
```

4.3 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

4.4 **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.

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}.

4.5 **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.

Referencias

Para la parte 1:

- Cap. 20 de R for Data Science (2e), de Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund.

Para el resto:

- Cap. 26-27 de R for Data Science (2e), de Hadley Wickham, Mine Çetinkaya-Rundel, and Garrett Grolemund.
- Cap. 11 de Hands-On Programming with R, de Garrett Grolemund.
- Sobre el ejercicio 4.3: 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.