EJERCICIO 02 - Estadistica Computacional

Byron Caices Lima

2023-10-05

---

1. Instale los paquetes gtools y combinat con las sentencias:

```
(library(gtools))
```

```
## [1] "gtools"    "stats"     "graphics"  "grDevices" "utils"     "datasets"
## [7] "methods"   "base"
```

```
(library(combinat))
```

```
##
## Attaching package: 'combinat'

## The following object is masked from 'package:utils':
##
##     combn

## [1] "combinat"  "gtools"    "stats"     "graphics"  "grDevices" "utils"
## [7] "datasets"  "methods"   "base"
```

---

2. Defina los siguientes conceptos y proporcione las fórmulas para calcularlos:

Tabla de definición:

Concepto	Toma Todos los Elementos	Orden Importante	Con Repetición
Permutación sin Repetición	No	Sí	No
Combinación sin Repetición	No	No	No
Permutación Circular	Sí	Sí	No
Permutación con Repetición	No	Sí	Sí
Combinación con Repetición	No	No	Sí

- Permutación sin Repetición

$$nPr = \frac{n!}{(n-r)!} \tag{1}$$

- Combinación sin Repetición

$$C(n,r) = \frac{n!}{r!(n-r)!} \tag{2}$$

- Permutación Circular

$$PCn = (n-1)! \tag{3}$$

- Permutación con Repetición

$$C(n,r) = n^r \tag{4}$$

- Combinación con Repetición

$$C(n,r) = \frac{(n+r-1)!}{r!(n-1)!} \tag{5}$$

---

3. Busque en la ayuda de R las funciones combinations, permutations del paquete gtools y permn, combn del paquete combinat. Explique brevemente cómo funcionan y qué diferencias hay entre ellas.

A. Definición de `combinations`: La función `combinations` del paquete `gtools` en R genera todas las posibles combinaciones de `n` elementos tomados de `r` en `r`.

Parámetros de la función `COMBINATIONS`:

- `n`: Número total de elementos.
- `r`: Número de elementos en cada combinación.
- `v`: Un vector con los `n` elementos a combinar. Si no se especifica, la función asume que los elementos son los números enteros de 1 a `n`.
- `repeats.allowed`: Un parámetro lógico. Si es `TRUE`, se permiten repeticiones en las combinaciones. Por defecto, es `FALSE`.

Ejemplo: Combinación S/R: Obtener todas las posibles combinaciones de 3 letras tomadas de un conjunto de 4 letras ("A", "B", "C", "D"):

```r
library(gtools)
result = combinations(n = 4, r = 3, v = c("A", "B", "C", "D"))
print(result)
```

```
##      [,1] [,2] [,3]
## [1,] "A"  "B"  "C"
## [2,] "A"  "B"  "D"
## [3,] "A"  "C"  "D"
## [4,] "B"  "C"  "D"
```

Ejemplo: Combinación C/R:

```
result_with_repeats = combinations(n = 5, r = 2, v = c("A", "B", "C", "D", "E"), repeats.allowed = TRUE)
print(result_with_repeats)
```

```
##       [,1] [,2]
##  [1,] "A"  "A"
##  [2,] "A"  "B"
##  [3,] "A"  "C"
##  [4,] "A"  "D"
##  [5,] "A"  "E"
##  [6,] "B"  "B"
##  [7,] "B"  "C"
##  [8,] "B"  "D"
##  [9,] "B"  "E"
## [10,] "C"  "C"
## [11,] "C"  "D"
## [12,] "C"  "E"
## [13,] "D"  "D"
## [14,] "D"  "E"
## [15,] "E"  "E"
```

B. Definición de `PERMUTATIONS`: La función `permutations` del paquete `gtools` en R genera todas las posibles permutaciones de `n` elementos tomados de `r` en `r`. A diferencia de las combinaciones, en las permutaciones el orden de los elementos es relevante.

Parámetros de la función `permutations`:

- `n`: Número total de elementos.
- `r`: Número de elementos en cada permutación.
- `v`: Un vector con los `n` elementos a permutar. Si no se especifica, la función asume que los elementos son los números enteros de 1 a `n`.
- `repeats.allowed`: Un parámetro lógico. Si es `TRUE`, se permiten repeticiones en las permutaciones. Por defecto, es `FALSE`.

Ejemplo de uso: Supongamos que deseas obtener todas las posibles permutaciones de 3 letras tomadas de un conjunto de 4 letras (“A”, “B”, “C”, “D”):

```
library(gtools)
result = permutations(n = 4, r = 3, v = c("A", "B", "C", "D"))
print(result)
```

```
##       [,1] [,2] [,3]
##  [1,] "A"  "B"  "C"
```

```
##  [2,] "A"  "B"  "D"
##  [3,] "A"  "C"  "B"
##  [4,] "A"  "C"  "D"
##  [5,] "A"  "D"  "B"
##  [6,] "A"  "D"  "C"
##  [7,] "B"  "A"  "C"
##  [8,] "B"  "A"  "D"
##  [9,] "B"  "C"  "A"
## [10,] "B"  "C"  "D"
## [11,] "B"  "D"  "A"
## [12,] "B"  "D"  "C"
## [13,] "C"  "A"  "B"
## [14,] "C"  "A"  "D"
## [15,] "C"  "B"  "A"
## [16,] "C"  "B"  "D"
## [17,] "C"  "D"  "A"
## [18,] "C"  "D"  "B"
## [19,] "D"  "A"  "B"
## [20,] "D"  "A"  "C"
## [21,] "D"  "B"  "A"
## [22,] "D"  "B"  "C"
## [23,] "D"  "C"  "A"
## [24,] "D"  "C"  "B"
```

Este código entrega una matriz con todas las permutaciones posibles de 3 letras tomadas de las 4 proporcionadas. Dado que el orden importa en las permutaciones, van a haber secuencias como "ABC", "ACB", "BCA", etc.

C. Definición de `PERMN`: La función `permn` del paquete `combinat` en R genera todas las posibles permutaciones de un conjunto dado.

Parámetros de la función `permn`:

- `x`: Un vector de elementos que se desea permutar.

Ejemplo de uso: Para obtener todas las permutaciones posibles del vector `c(1, 2, 3)`:

```r
result = permn(c(1, 2, 3))
print(result)
```

```
## [[1]]
## [1] 1 2 3
##
## [[2]]
## [1] 1 3 2
##
## [[3]]
## [1] 3 1 2
##
## [[4]]
```

```
## [1] 3 2 1
##
## [[5]]
## [1] 2 3 1
##
## [[6]]
## [1] 2 1 3
```

D. Definición de `COMBN`: La función `combn` genera todas las combinaciones posibles de un vector de elementos, tomados `m` a la vez.

Parámetros de la función `combn`:

- `x`: Un vector numérico o caracter de elementos que se desea combinar.
- `m`: El número de elementos en cada combinación.

Ejemplo de uso: Para obtener todas las combinaciones posibles de 2 elementos del vector `c(1, 2, 3)`:

```
result = combn(c(1, 2, 3), 2)
print(result)
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    2
## [2,]    2    3    3
```

Esto proporcionará una matriz donde cada columna representa una combinación única de 2 elementos tomados del vector `c(1, 2, 3)`. Específicamente, va a entregar las combinaciones `(1,2)`, `(1,3)` y `(2,3)`.

---

4. Calcule:

```
a_sin_rep = permutations(n = 9, r = 4)
cat("a) Sin repetición:", nrow(a_sin_rep), "\n")
```

a) Cantidad de permutaciones con n = 9 y r = 4 con y sin repetición.

```
## a) Sin repetición: 3024
```

```
a_con_rep = 9^4
cat("a) Con repetición:", a_con_rep, "\n")
```

```
## a) Con repetición: 6561
```

```r
b_sin_rep = combn(c("f", "g", "h", "i", "j"), 4)
cat("b) Sin repetición:\n")
```

b) Combinaciones de largo 4 con las letras f, g, h, i y j con y sin repetición.

```
## b) Sin repetición:
```

```r
print(b_sin_rep)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "f"  "f"  "f"  "f"  "g"
## [2,] "g"  "g"  "g"  "h"  "h"
## [3,] "h"  "h"  "i"  "i"  "i"
## [4,] "i"  "j"  "j"  "j"  "j"
```

```r
b_con_rep = combinations(n = 5, r = 4, v = c("f", "g", "h", "i", "j"), repeats.allowed = TRUE)
cat("b) Con repetición:\n")
```

```
## b) Con repetición:
```

```r
print(b_con_rep)
```

```
##       [,1] [,2] [,3] [,4]
##  [1,] "f"  "f"  "f"  "f"
##  [2,] "f"  "f"  "f"  "g"
##  [3,] "f"  "f"  "f"  "h"
##  [4,] "f"  "f"  "f"  "i"
##  [5,] "f"  "f"  "f"  "j"
##  [6,] "f"  "f"  "g"  "g"
##  [7,] "f"  "f"  "g"  "h"
##  [8,] "f"  "f"  "g"  "i"
##  [9,] "f"  "f"  "g"  "j"
## [10,] "f"  "f"  "h"  "h"
## [11,] "f"  "f"  "h"  "i"
## [12,] "f"  "f"  "h"  "j"
## [13,] "f"  "f"  "i"  "i"
## [14,] "f"  "f"  "i"  "j"
## [15,] "f"  "f"  "j"  "j"
## [16,] "f"  "g"  "g"  "g"
## [17,] "f"  "g"  "g"  "h"
## [18,] "f"  "g"  "g"  "i"
## [19,] "f"  "g"  "g"  "j"
## [20,] "f"  "g"  "h"  "h"
## [21,] "f"  "g"  "h"  "i"
## [22,] "f"  "g"  "h"  "j"
## [23,] "f"  "g"  "i"  "i"
## [24,] "f"  "g"  "i"  "j"
## [25,] "f"  "g"  "j"  "j"
## [26,] "f"  "h"  "h"  "h"
```

```
## [27,] "f"  "h"  "h"  "i"
## [28,] "f"  "h"  "h"  "j"
## [29,] "f"  "h"  "i"  "i"
## [30,] "f"  "h"  "i"  "j"
## [31,] "f"  "h"  "j"  "j"
## [32,] "f"  "i"  "i"  "i"
## [33,] "f"  "i"  "i"  "j"
## [34,] "f"  "i"  "j"  "j"
## [35,] "f"  "j"  "j"  "j"
## [36,] "g"  "g"  "g"  "g"
## [37,] "g"  "g"  "g"  "h"
## [38,] "g"  "g"  "g"  "i"
## [39,] "g"  "g"  "g"  "j"
## [40,] "g"  "g"  "h"  "h"
## [41,] "g"  "g"  "h"  "i"
## [42,] "g"  "g"  "h"  "j"
## [43,] "g"  "g"  "i"  "i"
## [44,] "g"  "g"  "i"  "j"
## [45,] "g"  "g"  "j"  "j"
## [46,] "g"  "h"  "h"  "h"
## [47,] "g"  "h"  "h"  "i"
## [48,] "g"  "h"  "h"  "j"
## [49,] "g"  "h"  "i"  "i"
## [50,] "g"  "h"  "i"  "j"
## [51,] "g"  "h"  "j"  "j"
## [52,] "g"  "i"  "i"  "i"
## [53,] "g"  "i"  "i"  "j"
## [54,] "g"  "i"  "j"  "j"
## [55,] "g"  "j"  "j"  "j"
## [56,] "h"  "h"  "h"  "h"
## [57,] "h"  "h"  "h"  "i"
## [58,] "h"  "h"  "h"  "j"
## [59,] "h"  "h"  "i"  "i"
## [60,] "h"  "h"  "i"  "j"
## [61,] "h"  "h"  "j"  "j"
## [62,] "h"  "i"  "i"  "i"
## [63,] "h"  "i"  "i"  "j"
## [64,] "h"  "i"  "j"  "j"
## [65,] "h"  "j"  "j"  "j"
## [66,] "i"  "i"  "i"  "i"
## [67,] "i"  "i"  "i"  "j"
## [68,] "i"  "i"  "j"  "j"
## [69,] "i"  "j"  "j"  "j"
## [70,] "j"  "j"  "j"  "j"
```

```
n = 30
r = 20

c_permutations = factorial(n) / factorial(n-r)

c_combinations = choose(n,r)
```

```r
cat("c) Permutación Sin repetición:", c_permutations, "\n")
```

c) Cantidad de permutaciones y combinaciones con n=30 y r=20 sin repetición.

```
## c) Permutación Sin repetición: 7.309658e+25
```

```r
cat("c) Permutación Sin repetición:", c_combinations, "\n")
```

```
## c) Permutación Sin repetición: 30045015
```

```r
d_sin_rep = permutations(n = 7, r = 5)
cat("d) Sin repetición:", nrow(d_sin_rep), "\n")
```

d) Cantidad de permutaciones con n=7 y r=5 con y sin repetición.

```
## d) Sin repetición: 2520
```

```r
d_con_rep = 7^5
cat("d) Con repetición:", d_con_rep, "\n")
```

```
## d) Con repetición: 16807
```

```r
e_sin_rep = combn(1:5, 2)
cat("e) Sin repetición:\n")
```

e) Combinaciones de largo 2 con los números 1, 2, 3, 4 y 5 con y sin repetición.

```
## e) Sin repetición:
```

```r
print(e_sin_rep)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1    1    1    1    2    2    2    3    3     4
## [2,]    2    3    4    5    3    4    5    4    5     5
```

```r
e_con_rep = combinations(n = 5, r = 2, v = 1:5, repeats.allowed = TRUE)
cat("e) Con repetición:\n")
```

```
## e) Con repetición:
```

```r
print(e_con_rep)
```

```
##       [,1] [,2]
##  [1,]    1    1
##  [2,]    1    2
##  [3,]    1    3
##  [4,]    1    4
##  [5,]    1    5
##  [6,]    2    2
##  [7,]    2    3
##  [8,]    2    4
##  [9,]    2    5
## [10,]    3    3
## [11,]    3    4
## [12,]    3    5
## [13,]    4    4
## [14,]    4    5
## [15,]    5    5
```

```
n = 50
r = 10

f_permutations = factorial(n) / factorial(n-r)

f_combinations = choose(n,r)

cat("f) Permutación Sin repetición:", c_permutations, "\n")
```

f) Cantidad de permutaciones y combinaciones con n=50 y r=10 sin repetición.

```
## f) Permutación Sin repetición: 7.309658e+25
```

```
cat("f) Permutación Sin repetición:", c_combinations, "\n")
```

```
## f) Permutación Sin repetición: 30045015
```

```
g_circular = factorial(6 - 1)
cat("g) Permutaciones circulares:", g_circular, "\n")
```

g) Cantidad de permutaciones circulares posibles con n=6.

```
## g) Permutaciones circulares: 120
```

```
h_sin_rep = combn(c("x", "y", "z", "w", "q"), 3)
cat("h) Sin repetición:\n")
```

h) Combinaciones de largo 3 con las letras x, y, z, w y q con y sin repetición.

```
## h) Sin repetición:
```

```
print(h_sin_rep)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,] "x"  "x"  "x"  "x"  "x"  "x"  "y"  "y"  "y"  "z"
## [2,] "y"  "y"  "y"  "z"  "z"  "w"  "z"  "z"  "w"  "w"
## [3,] "z"  "w"  "q"  "w"  "q"  "q"  "w"  "q"  "q"  "q"
```

```
h_con_rep = combinations(n = 5, r = 3, v = c("x", "y", "z", "w", "q"), repeats.allowed = TRUE)
cat("h) Con repetición:\n")
```

```
## h) Con repetición:
```

```
print(h_con_rep)
```

```
##       [,1] [,2] [,3]
##  [1,] "q"  "q"  "q"
##  [2,] "q"  "q"  "w"
##  [3,] "q"  "q"  "x"
##  [4,] "q"  "q"  "y"
##  [5,] "q"  "q"  "z"
##  [6,] "q"  "w"  "w"
##  [7,] "q"  "w"  "x"
##  [8,] "q"  "w"  "y"
##  [9,] "q"  "w"  "z"
## [10,] "q"  "x"  "x"
## [11,] "q"  "x"  "y"
## [12,] "q"  "x"  "z"
## [13,] "q"  "y"  "y"
## [14,] "q"  "y"  "z"
## [15,] "q"  "z"  "z"
## [16,] "w"  "w"  "w"
## [17,] "w"  "w"  "x"
## [18,] "w"  "w"  "y"
## [19,] "w"  "w"  "z"
## [20,] "w"  "x"  "x"
## [21,] "w"  "x"  "y"
## [22,] "w"  "x"  "z"
## [23,] "w"  "y"  "y"
## [24,] "w"  "y"  "z"
## [25,] "w"  "z"  "z"
## [26,] "x"  "x"  "x"
## [27,] "x"  "x"  "y"
## [28,] "x"  "x"  "z"
## [29,] "x"  "y"  "y"
## [30,] "x"  "y"  "z"
## [31,] "x"  "z"  "z"
## [32,] "y"  "y"  "y"
## [33,] "y"  "y"  "z"
## [34,] "y"  "z"  "z"
## [35,] "z"  "z"  "z"
```

5. Considere un problema de un repartidor que debe entregar paquetes en 40 casas y volver al origen sin pasar dos veces por la misma casa.

a) ¿Cuál es la probabilidad de que elija la ruta más corta? ¿Y la más larga? Para un repartidor que debe entregar paquetes en 40 casas, el número de rutas posibles sin repetir ninguna casa es simplemente **40!**.

Dado que hay una única ruta más corta y una única ruta más larga, la probabilidad de elegir cualquiera de estas rutas al azar es:

$$P(\text{ruta más corta o más larga}) = \frac{1}{40!}$$

b) Si elige una ruta al azar que es distinta a la anterior, ¿cuál es la probabilidad de que sea la ruta más corta? Si ya ha elegido una ruta, y ahora elige otra al azar, el número total de rutas posibles ahora es **40! - 1**, ya que hay una ruta menos para considerar (la ruta que ya eligió anteriormente). Sin embargo, aún hay una única ruta más corta, por lo que la probabilidad de elegir la ruta más corta en esta segunda elección es:

$$P(\text{ruta más corta en segunda elección}) = \frac{1}{40! - 1}$$

ADVERTENCIA: *"Las funciones* `combinations` *y* `permutations` *podrían no funcionar para este problema"*

Debido al tamaño extremadamente grande de los cálculos. El valor de **40!** es un número muy grande y calcular todas las permutaciones posibles de 40 elementos puede no ser práctico y consumiría una cantidad muy grande de memoria. Además para este tipo de problemas no es necesario generar todas las permutaciones o combinaciones explícitamente. En lugar de eso, se usaron fórmulas matemáticas para calcular las probabilidades.

---

6. Un restaurante tiene 4 camareros que deben atender a los clientes. Ana atiende al 20 % de los clientes y olvida 1 de cada 15 órdenes; Bob atiende al 50 % de los clientes y olvida 1 de cada 8 órdenes; Carlos atiende al 20 % de los clientes y olvida 1 de cada 12 órdenes; Diana atiende al 10 % de los clientes y olvida 2 de cada 7 órdenes.

a) (5 puntos) Formule la ecuación para resolver este problema. Para resolver las preguntas usaré el Teorema de Bayes, primero definiré unos eventos:

Eventos:

- A: El cliente es atendido por Ana.
- B: El cliente es atendido por Bob.
- C: El cliente es atendido por Carlos.
- D: El cliente es atendido por Diana.
- O: El cliente tiene una orden olvidada.
- $\neg O$: El cliente no tiene una orden olvidada.

Probabilidades dadas:

```r
P_A = 0.20
P_B = 0.50
P_C = 0.20
P_D = 0.10

P_O_dado_A = 1/15
P_O_dado_B = 1/8
P_O_dado_C = 1/12
P_O_dado_D = 2/7
```

A partir de las probabilidades dadas, podemos calcular las probabilidades complementarias de que no se olvide una orden dado un camarero:

```r
P_not_O_dado_A = 1 - P_O_dado_A
P_not_O_dado_B = 1 - P_O_dado_B
P_not_O_dado_C = 1 - P_O_dado_C
P_not_O_dado_D = 1 - P_O_dado_D
```

La probabilidad total de que un cliente no tenga una orden olvidada, considerando todos los camareros, es:

```r
P_not_O = P_A * P_not_O_dado_A + P_B * P_not_O_dado_B + P_C * P_not_O_dado_C + P_D * P_not_O_dado_D
```

Luego, una fórmula para la probabilidad de que un cliente haya sido atendido por `< camarero >` dado que su orden **no** fue olvidada es:

$$P(Camarero|\neg O) = \frac{P(\neg O|Camarero) \times P(Camarero)}{P(\neg O)}$$

b) (5 puntos) ¿Cuál es la probabilidad de que el cliente haya sido atendido por Carlos? Usando el Teorema de Bayes:

```r
# Probabilidad de que el cliente haya sido atendido por Carlos dado que no se olvidó su orden
P_C_dado_not_O = (P_not_O_dado_C * P_C) / P_not_O
cat("b) P(C|not O):", P_C_dado_not_O, "\n")
```

```
## b) P(C|not O): 0.2085873
```

```r
# Probabilidad de que el cliente haya sido atendido por Ana o Diana dado que no se olvidó su orden
P_A_dado_not_O = (P_not_O_dado_A * P_A) / P_not_O
P_D_dado_not_O = (P_not_O_dado_D * P_D) / P_not_O
P_A_or_D_dado_not_O = P_A_dado_not_O + P_D_dado_not_O
cat("c) P(A or D|not O):", P_A_or_D_dado_not_O, "\n")
```

c) (5 puntos) ¿Cuál es la probabilidad de que el cliente haya sido atendido por Ana o Diana?

```
## c) P(A or D|not O): 0.2936476
```

d) (5 puntos) Calcule la probabilidad de que el cliente haya sido atendido por cualquiera de los camareros. ¿Qué observa? Dado que uno de los camareros definitivamente atendió al cliente, la suma de las probabilidades de que el cliente haya sido atendido por cualquier camarero, independientemente de si se olvidó la orden o no, es:

```
P_cualquier_camarero = P_A + P_B + P_C + P_D
cat("d) P(cualquier camarero):", P_cualquier_camarero, "\n") # Esto debe ser 1
```

```
## d) P(cualquier camarero): 1
```

---

7. (10 puntos) De un grupo de 35 personas, se quiere conocer la opinión de 4 personas (elegidas al azar) sobre la legalización de la marihuana. Si se sabe que 18 personas están a favor y 17 en contra, ¿cuál es la probabilidad de que las 4 personas seleccionadas estén en contra?

Dado que hay 17 personas en contra, la probabilidad de seleccionar la primera persona en contra es $\frac{17}{35}$. Después de seleccionar una persona en contra, quedan 16 en contra de un total de 34 personas, por lo que la probabilidad de seleccionar la segunda persona en contra es $\frac{16}{34}$, y así sucesivamente.

La probabilidad de que las 4 personas seleccionadas estén en contra es:

$$P(\text{4 en contra}) = \frac{17}{35} \times \frac{16}{34} \times \frac{15}{33} \times \frac{14}{32}$$

```
prob_4_contra = (17/35) * (16/34) * (15/33) * (14/32)
prob_4_contra
```

```
## [1] 0.04545455
```
