Tarea 2

Luis Hernandez,Estefan Reyes,Andrés Ortega

2025-03-21

Ejercicio 8

Considera una urna con bolas de K diferente colores, donde al tiempo 0, hay α_i bolas del i-ésimo color. Al timepo n, se saca una bola y se regresa a la urna junto con una nueva bola de ese color y se repite el proceso N veces. Si $N \to \infty$, se puede domstra que las proporciones de bolas en la urna seguirán una distribución Dirichlet.

i. Considerando $K = 3$, $\alpha = c(2, 5, 1)$ y N “suficientemente grande”, implementa este algoritmo y obtén una muestra de tamaño n.

```r
# creamos una función que arroje las proporciones de la urna, y al ser bolas identicas esta tamb
ien será la probabilidad de sacar la bola de alpha_i

prob <- function(al){

  p_alpha <- numeric(length(al))
  for (i in 1:length(al)){
    #tomamos la proporcion del numero de bolas alpha_i y el total
    p_alpha[i] <- al[i]/sum(al)
  }
  return(p_alpha)
}


urna <- function(al,N){
  #usando las probabilidades
  prob <- prob(al)

  #creamos una matriz para ir guardando las proporciones
  M <- matrix(0, nrow = 1 , ncol = length(al))

  for (i in 1:N){
    #tomamos una bola
    ext <- sample(1:length(al),size =1, prob = prob)
    #a esa bola que sacamos la devolvemos más una extra y tenemos un total de las bolas alpha_i
+ 1
    al[ext] <- al[ext] + 1
    # Recalculamos las probabilidades
    prob <- prob(al)

    #Agregamos nuestro resultado de proporciones a la matriz
    M <- rbind(M, prob)

  }
  #eliminamos la primera fila que esta llena de ceros
  return(M[-1,])
}

#tomamos los parametros para la función recién creada
N <- 100
alpha <- c(2, 5, 1)

matriz_dirichlet1 <- urna(alpha, N)
print(tail(matriz_dirichlet1,10))
```

```
##           [,1]      [,2]       [,3]
## prob 0.1313131 0.8080808 0.06060606
## prob 0.1400000 0.8000000 0.06000000
## prob 0.1386139 0.8019802 0.05940594
## prob 0.1372549 0.8039216 0.05882353
## prob 0.1359223 0.7961165 0.06796117
## prob 0.1442308 0.7884615 0.06730769
## prob 0.1428571 0.7904762 0.06666667
## prob 0.1415094 0.7924528 0.06603774
## prob 0.1401869 0.7943925 0.06542056
## prob 0.1388889 0.7962963 0.06481481
```

Para este algoritmo resulta demasiado variable el resultado pues algunas tiradas no se parece a la distribución dirichlet en cuanto a sus esperanza, pero en la mayoría de los casos si lo hace; será necesario tomar muestras mucho más grandes.

ii. Diseña e implementa un algoritmo para genera vectores aleatorios Dirichlet a partir de la transformación de variables aleatorios Dirichlet a partir de la transformación de variables aletorias gamma. Obtén una muestra de tamaño n.

```r
# Función para generar vectores Dirichlet a partir de variables Gamma
vector_dirichlet <- function(alpha, n) {
  # Número de parámetros
  k <- length(alpha)

  # Inicializar matriz para almacenar las muestras
  gammas <- matrix(0, nrow = n, ncol = k)

  # Generar variables aleatorias Gamma para cada parámetro de alpha
  for (i in 1:k) {
    gammas[, i] <- rgamma(n, shape = alpha[i], rate = 1)
  }

  # Normalizar las muestras Gamma para obtener muestras Dirichlet
  s_dirichlet <- gammas / rowSums(gammas)

  return(s_dirichlet)
}

alpha <- c(2,5, 1)
n <- 100
matriz_dirichlet2 <- vector_dirichlet(alpha, n)
print(tail(matriz_dirichlet2,10))
```

```
##             [,1]      [,2]       [,3]
##  [91,] 0.3639293 0.2278083 0.408262434
##  [92,] 0.3498822 0.5553419 0.094775882
##  [93,] 0.1242704 0.4773805 0.398349110
##  [94,] 0.1020452 0.6518414 0.246113411
##  [95,] 0.3307279 0.4076664 0.261605720
##  [96,] 0.5010279 0.2933110 0.205661114
##  [97,] 0.2643043 0.5760599 0.159635702
##  [98,] 0.2873819 0.6251877 0.087430405
##  [99,] 0.3549131 0.6364366 0.008650278
## [100,] 0.5158495 0.3800030 0.104147505
```

iii. Para diferentes valores de N y n compara la media, el segundo momento y la covarianza muestral contra los resultados teóricos obtenidos en el ejercicio 7. ¿Cuál algoritmo es más eficiente?

```r
alpha <- c(2,5, 1)
n <- 1000
N <- 1000
matriz_dirichlet1 <- urna(alpha, N)
matriz_dirichlet2 <- vector_dirichlet(alpha, n)

#medias

media_urna <- colMeans(matriz_dirichlet1)
media_gammas <- colMeans(matriz_dirichlet2)

# Creamos la función para calcular lam media población calculada en el ejercicio 7
media_pob <- function(al){
  m_pob <- rep(0, length(al))

  for (i in 1:length(al)){
    m_pob[i] <- al[i]/sum(al)
  }

  return(m_pob)
}

media_poblacion <- media_pob(alpha)

#Unimos las estadisticas en dataframe para comparar
medias <- data.frame(media_urna, media_gammas, media_poblacion)

#Segundo momento
var_urna <- apply(matriz_dirichlet1, 2, var)
var_gammas <- apply(matriz_dirichlet2, 2, var)

var_pob <- function(al){
  v_pob <- rep(0,length(al))

  for (i in 1:length(al)){
    v_pob[i] <- media_poblacion[i]*(1-media_poblacion[i])/(1+sum(al)) + media_poblacion[i]^2
  }
  return(v_pob)
}

var_poblacion <- var_pob(alpha)

segundos_m <- data.frame(var_urna, var_gammas, var_poblacion)


#Covarianzas
cov_urna <- cov(matriz_dirichlet1)

cov_gammas <- cov(matriz_dirichlet2)

cov_pob <- function(al){
  M_cov_pob <- matrix(0, nrow = length(al), ncol = length(al))
```

```
  for (i in 1:length(al)){
    for(j in 1:length(al)){

      if (i == j){
        M_cov_pob [i,j] <- var_poblacion[j]
      }
      else{
        M_cov_pob[i,j] <- -(al[i]*al[j])/((sum(al)^2)*(sum(al)+1))
      }


    }
  }
  return(M_cov_pob)
}

cov_poblacion <- cov_pob(alpha)

covs <- data.frame(cov_urna, cov_gammas, cov_poblacion)

print(medias)
```

```
##   media_urna media_gammas media_poblacion
## 1 0.25799755    0.2530622           0.250
## 2 0.65486448    0.6236930           0.625
## 3 0.08713797    0.1232448           0.125
```

```
print(segundos_m)
```

```
##       var_urna var_gammas var_poblacion
## 1 5.083518e-04 0.02109283    0.08333333
## 2 8.395710e-04 0.02619932    0.41666667
## 3 8.136179e-05 0.01111080    0.02777778
```

```
print(covs)
```

```
##              X1            X2            X3         X1.1         X2.1
## 1  0.0005083518 -0.0006332805  1.249287e-04  0.021092829 -0.018090673
## 2 -0.0006332805  0.0008395710 -2.062905e-04 -0.018090673  0.026199319
## 3  0.0001249287 -0.0002062905  8.136179e-05 -0.003002156 -0.008108647
##           X3.1         X1.2         X2.2         X3.2
## 1 -0.003002156  0.083333333 -0.017361111 -0.003472222
## 2 -0.008108647 -0.017361111  0.416666667 -0.008680556
## 3  0.011110803 -0.003472222 -0.008680556  0.027777778
```

Vemos que para las distribución basada en la gamma la muetra de 1000 es suficientemente precisa con las estadisticas poblacionales pero la distribución basada en las urnas se queda muy corta en cuanto al segundo momento y las covarianzas como ya observamos cuando creamos la función necesitaremos muestras más

grandes.

```
alpha <- c(2,5, 1)
n <- 1000
N <- 30000
matriz_dirichlet1 <- urna(alpha, N)
matriz_dirichlet2 <- vector_dirichlet(alpha, n)

#mediaa

media_urna <- colMeans(matriz_dirichlet1)
media_gammas <- colMeans(matriz_dirichlet2)
media_poblacion <- media_pob(alpha)

medias <- data.frame(media_urna, media_gammas, media_poblacion)

#Segundo momento
var_urna <- apply(matriz_dirichlet1, 2, var)
var_gammas <- apply(matriz_dirichlet2, 2, var)
var_poblacion <- var_pob(alpha)

segundos_m <- data.frame(var_urna, var_gammas, var_poblacion)


#Covarianzas
cov_urna <- cov(matriz_dirichlet1)
cov_gammas <- cov(matriz_dirichlet2)
cov_poblacion <- cov_pob(alpha)

covs <- data.frame(cov_urna, cov_gammas, cov_poblacion)

print(medias)
```

```
##   media_urna media_gammas media_poblacion
## 1  0.2003496    0.2509776           0.250
## 2  0.6629028    0.6258002           0.625
## 3  0.1367476    0.1232222           0.125
```

```
print(segundos_m)
```

```
##       var_urna var_gammas var_poblacion
## 1 3.608167e-05 0.02098922    0.08333333
## 2 4.447419e-05 0.02530674    0.41666667
## 3 1.629245e-05 0.01231908    0.02777778
```

```
print(covs)
```

```
##              X1            X2            X3         X1.1         X2.1
## 1  3.608167e-05 -3.213171e-05 -3.949962e-06  0.020989224 -0.016988443
## 2 -3.213171e-05  4.447419e-05 -1.234248e-05 -0.016988443  0.025306738
## 3 -3.949962e-06 -1.234248e-05  1.629245e-05 -0.004000782 -0.008318295
##           X3.1         X1.2         X2.2         X3.2
## 1 -0.004000782  0.083333333 -0.017361111 -0.003472222
## 2 -0.008318295 -0.017361111  0.416666667 -0.008680556
## 3  0.012319077 -0.003472222 -0.008680556  0.027777778
```

Vemos que al hacer crecer la N se aproxima a la media poblacional pero el tiempo de ejecución se dispara enormemente y aún las estadisticas del segundo momento y las covarianzas quedan muy alejadas de las estadisticas poblacionales, es mejor el algoritmo que hace uso de la función gamma, el primero es demasiado costoso computacionalmente.

Ejercicio 9

Considera la base de datos $cork.txt$ que contiene los pesos de corcho tomado de las 4 direcciones cardinales de 28 árboles. ¿Se puede asumir que los datos siguen una distribución normal multivariada?

```
# convertimos el archivo a datos en R para empezar el análisis
datos9 <- read.csv("C:/Users/andre/OneDrive/Documentos/Actuaria/Semestre_2025_2/AnálisisMultivar
iado/Tareas/Tarea1/cork.txt", header = TRUE, sep = " ")

# Realizar la prueba de Mardia
resultado_mardia <- mult.norm(datos9)$mult.test

print(resultado_mardia)
```

```
##           Beta-hat    kappa     p-val
## Skewness  4.013694 18.73057 0.5393960
## Kurtosis 21.346375 -1.01337 0.3108836
```

La prueba de Mardia arroja unos p-valores superiores a nuestro nivel de significancia estandár, por lo tanto no hay suficiente evidencia para concluir que los datos no siguen una distribución normal multivariada, es decir tomamos los datos como datos distribuidos con una normal multivariada.

Ejercicio 10

Para la base de datos $wine.txt$ asumir que las variables alcohol y malic acid siguen un distribución normal multivarida.

i. Probar la hipótesis nula de que el vino promedio difiera de 13.15 grados de alcohol y 2.5 unidades de ácido málico.

```
datos10 <- read.csv("C:/Users/andre/OneDrive/Documentos/Actuaria/Semestre_2025_2/AnálisisMultivariado/Tareas/Tarea1/wine.txt", header = TRUE, sep = ",")

#Dejamos solo las columnas con las que vamos a trabajar
datos10 <- datos10[,2:3]

# Sigma desconocida
mu_bar=apply(datos10,2,mean)
sigma_hat=cov(datos10);sigma_hat
```

```
##               Alcohol Malic_acid
## Alcohol    0.65906233 0.08561131
## Malic_acid 0.08561131 1.24801540
```

```
# Statistic
((176*178)/(2*177))*t(mu_bar-c(13.15, 2.5))%*%solve(sigma_hat)%*%(mu_bar-c(13.15, 2.5))
```

```
##          [,1]
## [1,] 4.485113
```

```
# ICSNP
HotellingsT2(datos10,mu=c(13.15,2.5))
```

```
##
##  Hotelling's one sample T2-test
##
## data:  datos10
## T.2 = 4.4851, df1 = 2, df2 = 176, p-value = 0.01259
## alternative hypothesis: true location is not equal to c(13.15,2.5)
```

```
# Critical value at alpha=.05
qf(.95,2,176)
```

```
## [1] 3.047307
```

Vemos que nuestro valor de la estadistica esta por encima de nuestro valor critico, y de la misma manera el p-valor es menor al nivel de significancia estandár por lo tanto rechazamos la hipotesis nula de las medias en el vector c(13.15, 2.5).

ii. Realizar los contrastes de hipótesis necesarios para verificar si existe o no una diferencia para los niveles de alcohol y ácido málico para las clases 1 y 2 de vinos.

```r
datos10 <- read.csv("C:/Users/andre/OneDrive/Documentos/Actuaria/Semestre_2025_2/AnálisisMultivariado/Tareas/Tarea1/wine.txt", header = TRUE, sep = ",")

#Dejamos solo las columnas con las que vamos a trabajar
datos10 <- datos10[,1:3]

datos10_1 <- datos10[datos10$Class == 1,]
datos10_2 <- datos10[datos10$Class == 2,]

datos10_1 <- datos10_1[,2:3]
datos10_2 <- datos10_2[,2:3]

# Combinar los datos en una lista
datos <- list(datos10_1, datos10_2)

# Realizar la prueba de Hotelling T^2
resultado <- hotelling.test(datos[[1]], datos[[2]])

# Mostrar los resultados
print(resultado)
```

```
## Test stat:  272.45
## Numerator df:  2
## Denominator df:  127
## P-value:  0
```

Tenemos que el p-valor para la prueba de hotelling es practicamente nulo, por lo tanto la hipotesis nula que dice que las medias son iguales para ambos grupos no se cumple y por tanto las medias para ambos grupos son diferentes.