1. Problemas

1.1. Resuelve los siguientes problemas cortos:

(a) Poincaré ha quedado con su amigo Grigori en Sol. Grigori suele retrasarse de media 5 minutos, pero nunca se retrasa más de 8 minutos. Teniendo en cuenta que Poincaré llegará 3 minutos tarde, ¿Qué probabilidad hay de que Grigori esté en Sol antes que Poincaré? Aplica una distribución uniforme al tiempo de llegada de Grigori.

(b) Si el 99 % de los correos no deseados contiene una palabra maliciosa, el 1 % de los correos deseados contiene palabras maliciosas y el 2 % de los correos recibidos son no deseados, ¿Qué probabilidades hay de que un correo electrónico con palabra maliciosa sea no deseado?

Solution: a) Consideramos la variable aleatoria X = "Tiempo de retraso de Grigori" y, tal como dice el enunciado, asumimos que $X \sim U(a, b)$. Como Grigori nunca se retrasa más de 8 minutos, entonces $b = 8$. Además, utilizando la fórmula de la esperanza:

$$5 = \frac{a+b}{2}$$

Obtenemos que $a = 2$. Calculamos la probabilidad con punif(3,2,8) $\approx$ 0,167.

b) Consideramos los eventos A = "correo no deseado" y B = "palabra maliciosa en el correo". Aplicamos Bayes (un árbol puede ayudar a realizar los cálculos):

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|A^c)P(A^c)} = \frac{0,99 \cdot 0,02}{0,99 \cdot 0,02 + 0,01 \cdot 0,98} \approx 0.67$$

1.2. Un sistema de procesamiento en la nube ejecuta dos tareas paralelas (X,Y) que deben completarse para que una transacción sea exitosa. Los tiempos de ejecución conjuntos de ambas tareas siguen una distribución con función de densidad:

$$f(x, y) = \begin{cases} 2e^{-(x+2y)} & \text{para } x, y \geq 0 \\ 0 & \text{en otro caso} \end{cases}$$

donde x e y se miden en segundos.

(a) Teniendo en cuenta que las tareas se ejecutan en paralelo, calcula la probabilidad de que una transacción se complete en menos de 2 segundos. Expresa matemáticamente esta probabilidad y resuélvela.

(b) Calcula la probabilidad de que la tarea X tarde al menos 1 segundo más que la tarea Y. Expresa matemáticamente esta probabilidad y resuélvela.

(c) Si sabes que la tarea Y ha tardado exactamente 3 segundos, ¿cuál es la probabilidad de que la tarea X tarde al menos 4 segundos? Expresa matemáticamente esta probabilidad usando la densidad condicionada y resuélvela.

Solution:

```
source("utils.R")

f_xy <- function(x, y) {
  ifelse(
```

```r
    (x > 0) & (y > 0),
    2 * exp(-(x + 2 * y)),
    0
  )
}

# P(X < 2, Y < 2)
integrate2_dydx(
  f_xy,
  0,
  2,
  function(x) 0,
  function(x) 2
)

# P(X > Y + 1)
integrate2_dydx(
  f_xy,
  -Inf,
  Inf,
  function(x) -Inf,
  function(x) x - 1
)

# P(X < 4|Y=3) needs f(x|y=3) = f(x, y=3) / fy(3)

marginal_y <- function(y) integrate(f_xy, -Inf, Inf, y = y)$value
f_given_y3 <- function(x) {
  f_xy(x, y = 3) / marginal_y(3)
}
# P(X < 4|Y=3) is
integrate(f_given_y3, -Inf, 4)
```

1.3. Se baraja una baraja española (40 cartas, 4 palos) antes de iniciar el siguiente juego. Un jugador extrae cartas continuamente según estas reglas:

- El jugador toma la primera carta del mazo
- Si la carta es del mismo palo que la anterior, la devuelve, baraja el mazo y continúa extrayendo
- Si es de distinto palo, la retira del mazo y continúa extrayendo

El jugador gana si logra quedarse sin cartas. Es imposible ganar si en algún momento solo quedan cartas del mismo palo en el mazo.

¿Qué probabilidad hay de ganar?

Pista: En R, puedes usar índices negativos para eliminar elementos de un vector. Ejemplo: `v = c(``a", ``b", ``c", ``d"); v[-2]`

Solution:

```r
baraja <- rep(c("A", "B", "C", "D"), each = 10)
baraja <- sample(baraja)

simulate_game <- function() {
  success <- TRUE
  previous <- "E"
  while(length(baraja) > 0) {
    # Siempre cogemos y quitamos la primera carta
    index <- 1
    carta <- baraja[[index]]
    # No coincide...
    if (carta != previous) {
      # ... por lo que quitamos la carta de la baraja
      baraja <- baraja[-index]
      previous <- carta
    } else {
      # Coincide, por lo que no quitamos y barajamos
      baraja <- sample(baraja)
    }
    n_palos <- length(table(baraja))
    if ((n_palos == 1) && (length(baraja) > 1)) {
      success <- FALSE
      break
    }
  }
  success
}

prob <- mean(
  replicate(5000, simulate_game())
)
print(prob)
```

1.4. Una aplicación web procesa peticiones que pueden ser de dos tipos: peticiones que requieren acceso a base de datos, con probabilidad 0.7; y peticiones que solo requieren procesamiento en memoria, con probabilidad 0.3. Si la petición requiere acceso a base de datos, el tiempo de respuesta se puede aproximar por una distribución Normal de media 200 milisegundos y varianza 900. Si la petición solo requiere procesamiento en memoria, el tiempo de respuesta se puede aproximar por una distribución Normal de media 150 milisegundos y varianza 400. Se pide:

(a) Escribe una función `simulate_request_time(n_sims)` que devuelva `n_sims` de las variables aleatorias relevantes.

(b) Usando simulaciones, calcula el tiempo medio de respuesta de una petición.

(c) Calcula la probabilidad de que una petición tarde más de 200 milisegundos en ser procesada usando simulaciones.

(d) Si observas que una petición ha tardado más de 160 milisegundos, ¿cuál es la probabilidad de que haya sido una petición que requería acceso a base de datos? Resuelve mediante simulaciones.

]

Solution:

```r
simulate_request_time <- function(n_sims=5000) {
  needs_db <- rbinom(n_sims, 1, 0.7)
  times <- ifelse(
    needs_db == 1,
    rnorm(n_sims, 200, sd = sqrt(900)),
    rnorm(n_sims, 150, sd = sqrt(400))
  )
  cbind("needs_db" = needs_db, "times" = times)
}

sims <- simulate_request_time()
needs_db <- sims[, "needs_db"]
times <- sims[, "times"]

# b)
mean(times)
# c)
mean(times > 200)
# d)
xxx <- 160
sum(
  (needs_db == 1) & (times > xxx)
) / sum(
  times > xxx
)
```