Actuarial 3 Tarea 3

Actuarial 3

Marcelino Sánchez

7/11/23

Ejercicio 2: Empalmes

Para un seguro de daños, se conoce que las pérdidas menores a c siguen una distribución $Exp(\frac{1}{\theta})$ con $\theta > 0$, y las pérdidas mayores o iguales a c tienen una distribución $Pareto(\alpha, \beta)$ donde $\alpha > 0$ y $\beta > 0$.

a) Se pide demostrar que la función de densidad de la Distribución de Empalme está dada por:

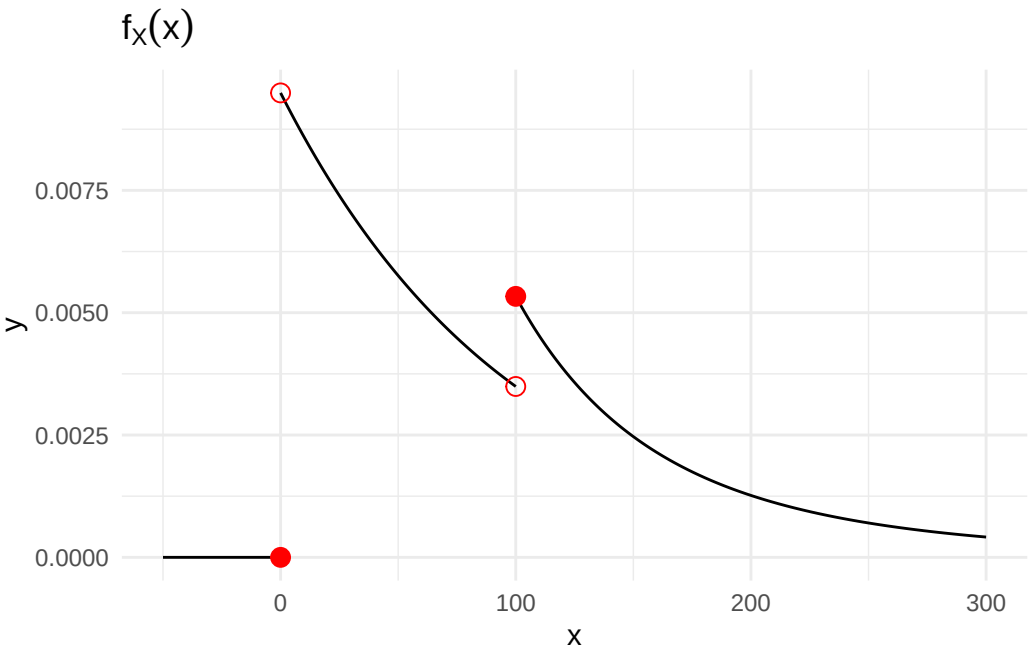
$f_X(x) = p\frac{\theta^{-1}e^{-\frac{x}{\theta}}}{1-e^{-\frac{c}{\theta}}}I(0 < x < c) + (1-p)\frac{\alpha(c+\beta)^{\alpha}}{(x+\beta)^{\alpha+1}}I(x \geq c)$

Dado:

- $\theta = 100$
- $\alpha = 4$
- $\beta = 200$
- $c = 100$

Se pide:

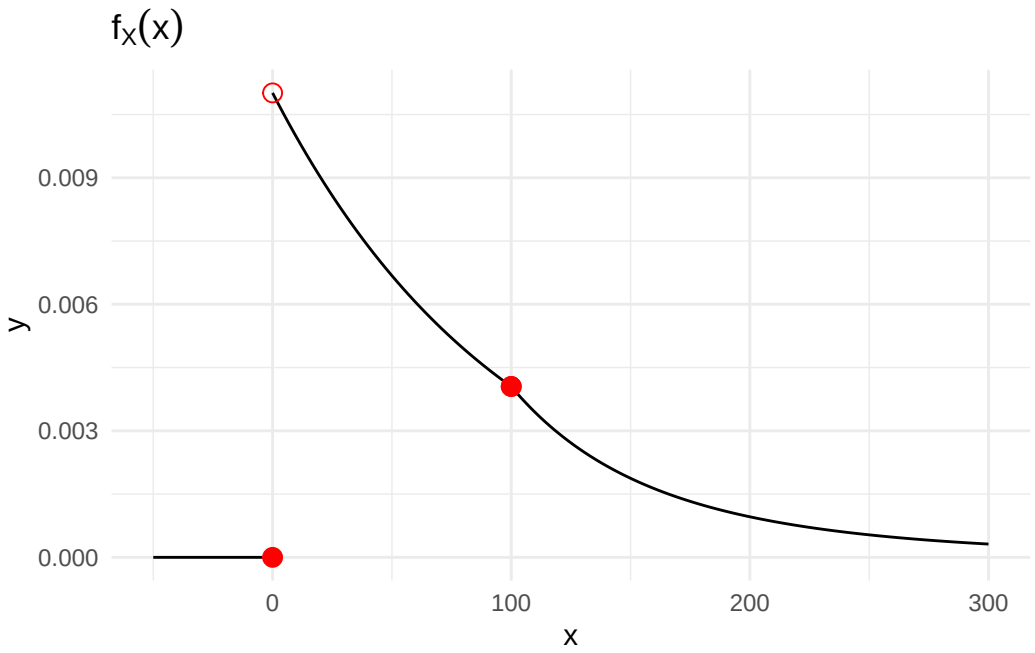
i) Graficar $f_X(x)$ con $p = 0.6$. Determinar si $f_X(x)$ es continua en $x = 100$.



ii) Encontrar el valor de $ p $ para que $ f_X(x) $ sea continua en $ x = 100 $ y luego graficar $ f_X(x) $ con este valor.

```
theta <- 100
alpha <- 4
beta <- 200
c <- 100
aux1 <- ((c+beta)*(theta^(-1))*((exp(-c/theta))/(1-exp(-c/theta)))
 + alpha)
pvalor <- alpha/aux1
```

El valor de p en el cual se satisface la continuidad en c es 0.6961449



Ejercicio 4: Colas de las distribuciones

Suponga que $X \sim \text{Pareto}(\alpha, \theta)$ y $Y \sim \text{Gamma}(\tau, \beta)$.

Comparación de distribuciones

b) Para comparar de manera más justa ambas distribuciones, queremos que ambas tengan media 5 y varianza 75.

c) Encuentre los valores de α, θ, τ, y β.

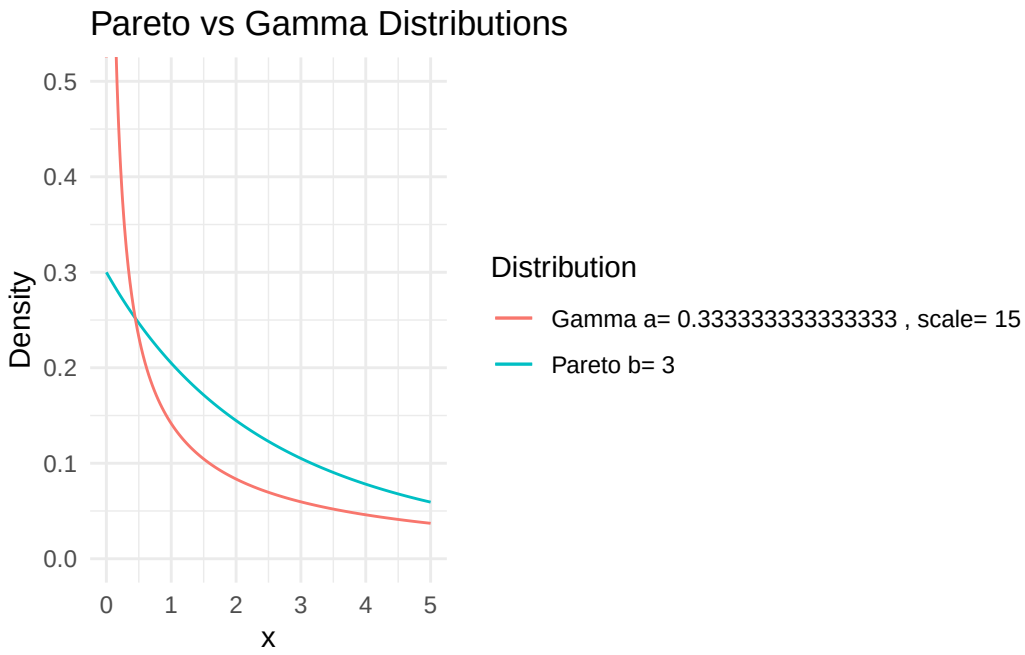
```r
#Calculamos los parámetros de la pareto
varianza <- 75

esperanza <- 5

aux1 <- varianza/(esperanza^2) +1

aux2 <- varianza/(esperanza^3) -1/esperanza

theta <- aux1/aux2
```

```r
alpha <- theta/esperanza + 1

#Calculamos los parámetros de la gamma

tau <- (esperanza^2)/varianza

beta <- varianza/esperanza
```

Los valores de los parámetros son:

- $\alpha = 3$
- $\theta = 10$
- $\tau = 0.3333333$
- $\beta = 15$

ii) Compare en la misma gráfica ambas funciones de densidad. ¿Qué conclusiones se pueden derivar?



Se puede derivar que la gamma acumula más que la pareto para valores cercanos a cero (aún teniendo la misma varianza y media), por lo que la pareto tiene cola más pesada que la gamma.

Ejercicio 5

Verosimilitud para datos agrupados. Suponga que las pérdidas de cierto riesgo siguen una distribución con función de distribución acumulada $F(x) = (1 - \frac{\theta}{x})I(\theta, \infty)(x)$, donde $\theta > 0$. Una muestra aleatoria de 20 siniestros contiene 9 pérdidas por debajo de 10 mil dólares, 6 pérdidas entre 10 y 25 mil dólares y 5 pérdidas que exceden los 25 mil dólares. Encuentre el estimador de máxima verosimilitud de θ. [10]

Por principio de máxima verosimilitud tenemos que el logaritmo de la función de verosimilitud para datos agrupados está dada por:

$$\ln(L(\theta)) = 9\ln\left(F(10000,\theta) - F(0,\theta)\right) + 6\ln\left(F(25000,\theta) - F(10000,\theta)\right)$$

$$+5\ln\left(1 - F(25000,\theta)\right)$$

```
acumulada <- function(x, theta){
  if(x > theta){
    resultado = 1 - (theta/x)
  }else{
    resultado = 0
  }
  return(resultado)
}


verosimilitud <- function(theta){
  resultado <- 9*log(acumulada(10000, theta)-acumulada(0,theta)) +
   6*log(acumulada(25000, theta)-acumulada(10000,theta)) +
   5*log(1-acumulada(25000,theta))
  return(resultado)
}

neg_verosimilitud <- function(theta) {
  return(-verosimilitud(theta))
}

# Usamos la función optimize para encontrar el
#valor de theta que maximiza la verosimilitud
# Establece un rango razonable para
#theta basado en tu conocimiento del problema
```

```r
resultado <- optimize(neg_verosimilitud, interval=c(0, 9000))
```

El valor de θ que maximiza la verosimilitud numéricamente es 5499.9999998 .

Ejercicio 6

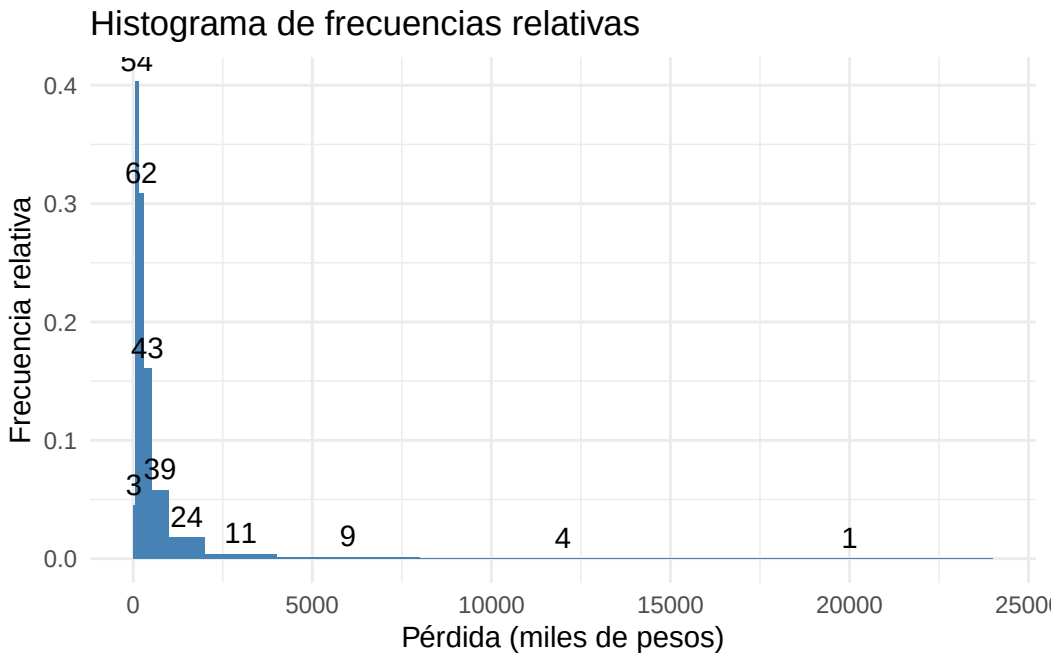
Ajuste para datos agrupados. A continuación se presenta la distribución de frecuencias de una muestra de 250 pérdidas de un seguro de hogar.

Pérdida (miles de pesos)	Frecuencia
0 - 50	3
50 - 150	54
150 - 300	62
300 - 500	43
500 - 1,000	39
1,000 - 2,000	24
2,000 - 4,000	11
4,000 - 8,000	9
8,000 - 16,000	4
Más de 16,000	1
Total	**250**

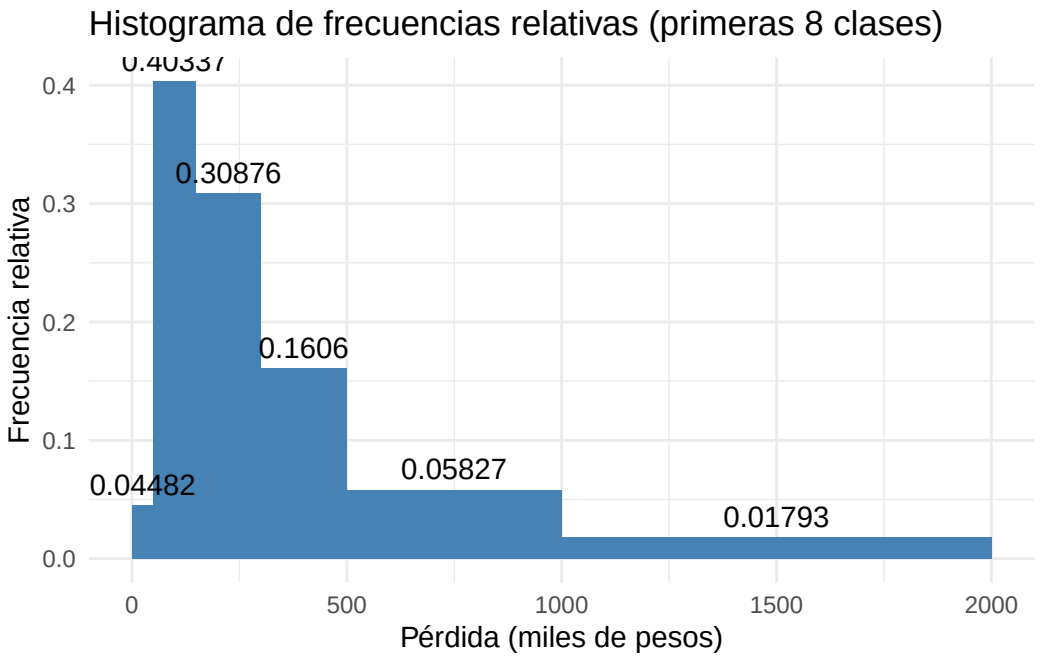
a) Construya el histograma de frecuencias relativas respectivo. Note que las clases son de distinta longitud. [10]

```r
# Crea un vector con los límites de las clases

clases <- c(0, 50, 150, 300, 500, 1000, 2000, 4000, 8000, 16000, 24000)

# Crea un vector con las frecuencias

frecuencias <- c(3, 54, 62, 43, 39, 24, 11, 9, 4, 1)

# Crea un vector con las frecuencias relativas

frecuencias_relativas <- frecuencias/sum(frecuencias)

# Crea un vector con las longitudes de las clases

longitudes_clases <- diff(clases)
```

```r
# Crea un vector con los factores de ajuste

factores_ajuste <- 1/longitudes_clases

# Crea un vector con las frecuencias relativas ajustadas

frecuencias_relativas_ajustadas <- frecuencias_relativas * factores_ajuste

# Crea un vector con las frecuencias relativas corregidas

frecuencias_relativas_corregidas <- frecuencias_relativas_ajustadas/sum(frecuencias_relati

# Crea un vector con los puntos medios de las clases

puntos_medios_clases <- (clases[-1] + clases[-length(clases)])/2

# Crea un data frame con los datos

df <- data.frame( frecuencias, frecuencias_relativas, frecuencias_relativas_ajustadas,
frecuencias_relativas_corregidas, puntos_medios_clases)


# Grafica el histograma
gg1 <-ggplot(df, aes(x = puntos_medios_clases, y = frecuencias_relativas_corregidas)) +
  geom_bar(width = longitudes_clases, stat = "identity", fill = "steelblue") +
  geom_text(aes(label = frecuencias), vjust = -0.5) +
  ggtitle("Histograma de frecuencias relativas") +
  xlab("Pérdida (miles de pesos)") +
  ylab("Frecuencia relativa") +
  theme_minimal()

gg1
```




```r
ggplot(df[1:6, ], aes(x = puntos_medios_clases, y = frecuencias_relativas_corregidas)) +
  geom_bar(width = longitudes_clases[1:6], stat = "identity", fill = "steelblue") +
  geom_text(aes(label = round(frecuencias_relativas_corregidas,digits = 5)), vjust = -0.5)
  ggtitle("Histograma de frecuencias relativas (primeras 8 clases)") +
  xlab("Pérdida (miles de pesos)") +
  ylab("Frecuencia relativa") +
  xlim(c(0, 2000)) +  # Limitar el eje x hasta un poco después de 8,000
  theme_minimal()
```



b) A partir de los datos agrupados, ajuste por máxima verosimilitud la Distribución Exponencial Inversa con función de distribución acumulada $F(x) = e^{-\frac{\theta}{x}} I(0,\infty)(x), \quad \theta > 0$. Especifique $\widehat{\theta}$ y grafique la densidad ajustada junto con el histograma del inciso anterior. [10]

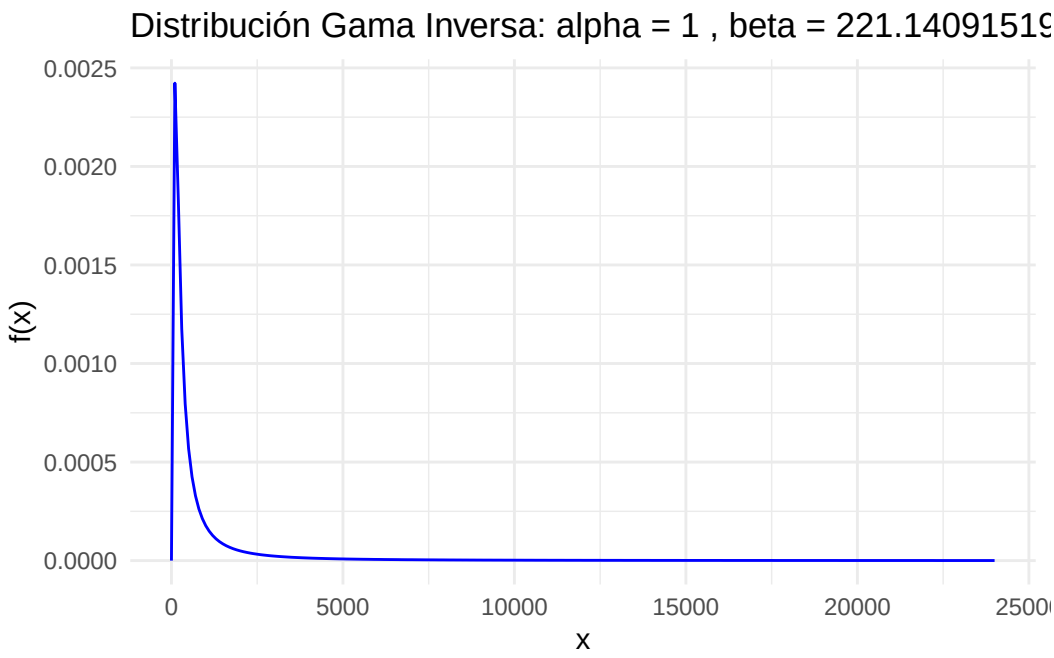
Por principio de máxima verosiimilitud tenemos que el logaritmo de la función de verosimilitud para datos agrupados está dada por:

$$\ln(L(\theta)) = \sum_{i=1}^{10} n_i \ln\left(F(b_i, \theta) - F(a_i, \theta)\right)$$

```
acumulada <- function(x, theta){
  if(x > 0){
    resultado = exp(-theta/x)
  }else{
    resultado = 0
  }
  return(resultado)
}


verosimilitud <- function(theta){
```

```r
  for(i in 1:10){
    if(i == 1){
      resultado <- frecuencias[i]*log(acumulada(clases[i+1], theta)-
      acumulada(clases[i],theta))
    }else{
      resultado <- resultado +
      frecuencias[i]*log(acumulada(clases[i+1], theta)-acumulada(clases[i],theta))
    }
  }
  return(resultado)
}

neg_verosimilitud <- function(theta) {
  return(-verosimilitud(theta))
}

# Usamos la función optimize para encontrar el valor de theta que maximiza la verosimilitu
# Establece un rango razonable para theta basado en tu conocimiento del problema
resultado <- optimize(neg_verosimilitud, interval=c(0, 9000))



# Compute likelihoods

d_gamma_inv <- function(x, theta) {
  ifelse(x > 0, (theta / (x^(2))) * exp(-theta/x),0)
}

# Valores de x
x_vals <- seq(0, 24000, by = 100)

# Calcular la función de densidad para estos valores
y_vals <- sapply(x_vals, d_gamma_inv, theta =resultado$minimum )

# Graficar

data_to_plot2 <- data.frame(x = x_vals, y = y_vals)

ggplot(data_to_plot2, aes(x = x, y = y)) +
  geom_line(color = "blue") +
  ggtitle(paste("Distribución Gama Inversa: alpha =", 1, ", beta =", resultado$minimum)) +
```

```
  xlab("x") +
  ylab("f(x)") +
  theme_minimal()
```




```
# Primero, configura el ggplot base
plot_base <- ggplot() +
  theme_minimal() +
  xlab("x") + ylab("f(x)")

# Supongo que plot_base y los otros datasets ya están definidos

# Calcula el factor de transformación. Este factor se basará en la
#relación entre los rangos máximos de tus datos.
max_y_data_to_plot2 <- max(data_to_plot2$y)
max_df <- max(df$frecuencias_relativas_corregidas)

# Factor de transformación
trans_factor <- max_y_data_to_plot2 / max_df

final_plot <- plot_base +
  # Primero traza el histograma
  geom_bar(data = df, aes(x = puntos_medios_clases, y = frecuencias_relativas_corregidas,
```

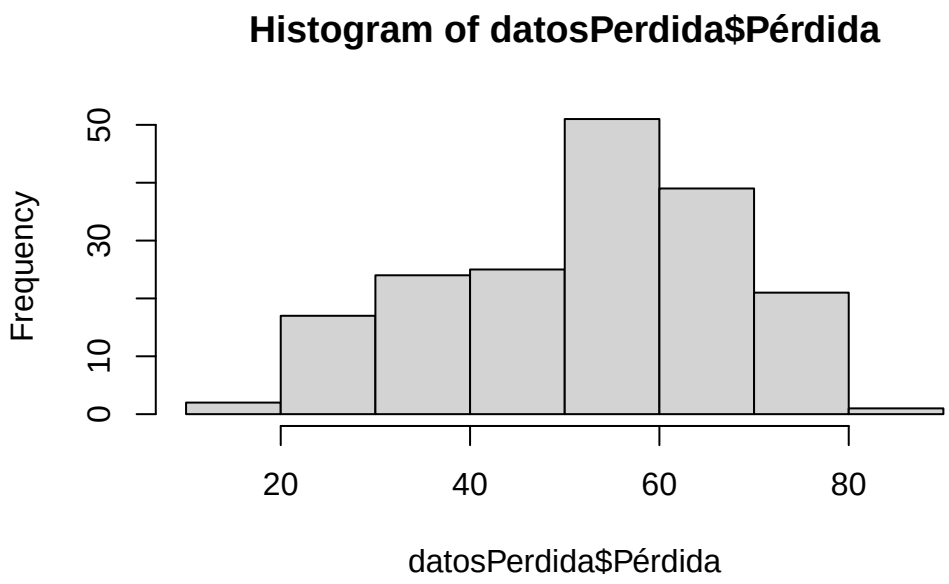
```
  width = longitudes_clases), stat = "identity", fill = "steelblue") +
  # Luego traza la línea, pero multiplica los valores por el factor de transformación
  geom_line(data = data_to_plot2, aes(x = x, y = y / trans_factor), color = "red") +
  ggtitle(paste("Distribución Gama Inversa: alpha =",
   1, ", beta =", resultado$minimum)) +
  # Añade el eje secundario, dividiendo por el
  #factor de transformación para mostrar los valores reales
  scale_y_continuous(sec.axis = sec_axis(~./trans_factor, name = "Eje secundario"))

# Finalmente, muestra el gráfico
print(final_plot)
```

El valor de θ que maximiza la verosimilitud numéricamente es 221.1409152 .

Ejercicio 7

Ajuste de una mezcla. Con base en 180 pérdidas de un seguro de vehículos que cubre automóviles y motocicletas se construyó el siguiente histograma:

```r
library(readxl)
library(kableExtra)
ruta <- paste0(getwd(), "/bases/Tarea 3.xlsx")
datosPerdida <- read_excel(ruta, sheet = "7", range = "A8:B188")

hist(datosPerdida$Pérdida)
```



Se quiere ajustar una mezcla de distribuciones Ji-Cuadradas con medias ν y 2ν con ponderadores α y $1-\alpha$, respectivamente, donde $\nu \in \mathbb{Z}^+$ y $0 < \alpha < 1$.

a) A partir de los datos individuales, escriba la función de densidad de la mezcla y verifique que los EMV son $\hat{\alpha} = 0.2468$ y $\hat{\nu} = 30$. (Sugerencia: Optimice numéricamente respecto a ambos parámetros, luego fije $\hat{\nu}$ al entero más cercano y maximice nuevamente para obtener $\hat{\alpha}$). [15]

Primero notamos que la función de verosimilitud está dada por:

$$L(\nu, \alpha) = \prod_{i=1}^{180} \left[\alpha \frac{x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)} + (1-\alpha) \frac{x^{\nu-1} e^{-\frac{x}{2}}}{2^{\nu} \Gamma(\nu)} \right]$$

Por lo que el logaritmo de la función de verosimilitud está dado por:

$$\ln(L(\nu, \alpha)) = \sum_{i=1}^{180} \left[\ln \left(\alpha \frac{x^{\frac{\nu}{2}-1} e^{-\frac{x}{2}}}{2^{\frac{\nu}{2}} \Gamma\left(\frac{\nu}{2}\right)} + (1-\alpha) \frac{x^{\nu-1} e^{-\frac{x}{2}}}{2^{\nu} \Gamma(\nu)} \right) \right]$$

```r
# Optimizamos numéricamente respecto a ambos parámetros

mezclaFuncion <- function(x, alpha, nu) {
  alpha * dgamma(x, shape = nu/2, scale = 2) + (1 - alpha) * dgamma(x,
  shape = nu, scale = 2)+.00000001
}

verosimilitud <- function(alpha, nu, datos) {
  log_mezcla <- log(mezclaFuncion(datos, alpha, nu))
  resultado <- sum(log_mezcla)
  return(resultado)
}

neg_verosimilitud <- function(params, datos) {
  alpha <- params[1]
  nu <- params[2]
  -verosimilitud(alpha, nu, datos)
}

# Suponiendo que tienes un dataframe llamado datosPerdida con una columna 'Pérdida'
datos <- datosPerdida$Pérdida

# Valores iniciales para alpha y nu
valores_iniciales <- c(alpha = 0.5, nu = 2)

# Límites para alpha y nu
# alpha está entre 0 y 1, y nu es mayor que 0
limites <- list(c(0.000001, 10), c(1e-6, 50))

# Utilizar optim con el método "L-BFGS-B" para encontrar los parámetros óptimos
resultado_optim <- optim(
  par = valores_iniciales,
  fn = neg_verosimilitud,
  datos = datos,
  method = "L-BFGS-B",
  lower = sapply(limites, `[`, 1), # Límites inferiores
  upper = sapply(limites, `[`, 2)  # Límites superiores
)

# Los parámetros optimizados
(alpha_optim <- resultado_optim$par[1])
```

```
    alpha
0.2428192
```

```r
(nu_optim <- resultado_optim$par[2])
```

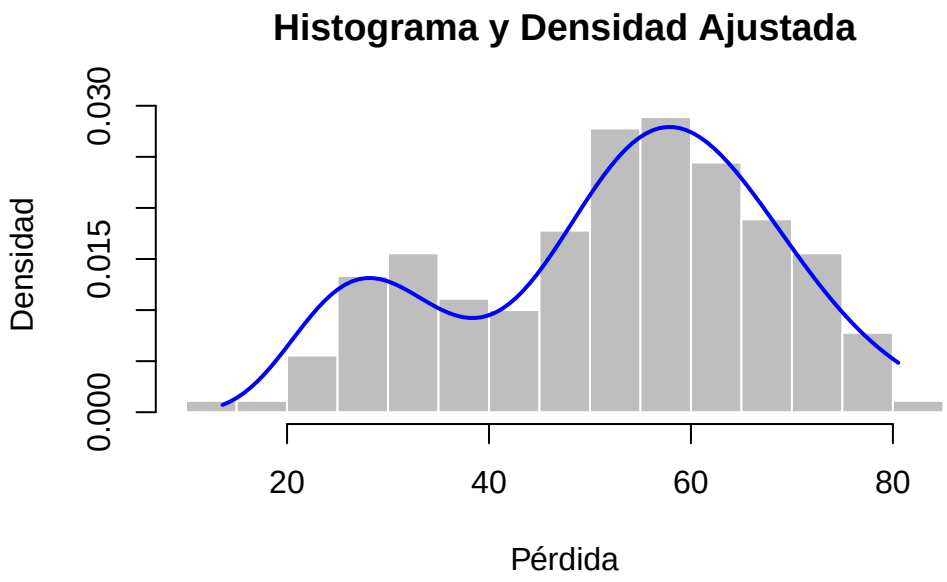
```
      nu
29.81585
```

```r
# Optimizamos numéricamente respecto a alpha

# Ahora neg_verosimilitud aceptará solo alpha ya que nu es fijo.
neg_verosimilitud <- function(alpha, datos) {
  nu <- 30  # Fijamos nu a 30
  -verosimilitud(alpha, nu, datos)
}

# Suponiendo que tienes un dataframe llamado datosPerdida con una columna 'Pérdida'
datos <- datosPerdida$Pérdida

# Valor inicial solo para alpha
valor_inicial_alpha <- 0.5

# Límite solo para alpha
limites_alpha <- c(0.000001, 1)

# Utilizar optim con el método "L-BFGS-B" para encontrar el parámetro óptimo de alpha
resultado_optim <- optim(
  par = valor_inicial_alpha,
  fn = neg_verosimilitud,
  datos = datos,
  method = "L-BFGS-B",
  lower = limites_alpha[1], # Límite inferior para alpha
  upper = limites_alpha[2]  # Límite superior para alpha
)

# El parámetro optimizado para alpha
alpha_optim <- resultado_optim$par
```

Con lo cual obtenemos que $\hat{\alpha} = 0.2468169$ y $\hat{\nu} = 30$.

b) Grafique la función de densidad ajustada junto con el histograma de frecuencias relativas y obtenga los EMV de las modas. [15]

```r
# Asumiendo que alpha_optim contiene el valor de alpha optimizado
alpha_emv <- alpha_optim
nu_emv <- 30  # Valor fijo de nu

# 1. Crear una secuencia de valores sobre los que queremos calcular la densidad ajustada
x_values <- seq(min(datosPerdida$Pérdida), max(datosPerdida$Pérdida), length.out = 100)

# 2. Calcular la función de densidad para cada uno de esos valores
densidad_ajustada <- mezclaFuncion(x_values, alpha_emv, nu_emv)

# 3. Crear el histograma con frecuencias relativas
hist(datosPerdida$Pérdida, breaks = 20, probability = TRUE,
col = 'gray', border = 'white',
     main = 'Histograma y Densidad Ajustada',
     xlab = 'Pérdida', ylab = 'Densidad')

# 4. Sobreponer la función de densidad ajustada
lines(x_values, densidad_ajustada, col = 'blue', lwd = 2)
```




```r
# 5. Obtener las modas

# Punto de inicio para la búsqueda de la moda,
#podría ser el valor medio de los datos, por ejemplo
valor_inicial1 <- 60
```

```r
valor_inicial2 <- 25
# Necesitamos una función que sea negativa para la
#función de densidad para poder minimizarla
neg_densidad_mezcla <- function(x, alpha, nu) {
  -mezclaFuncion(x, alpha, nu)
}
# Usar optim para encontrar la moda de la función de densidad
resultado_moda <- optim(
  par = valor_inicial1,
  fn = neg_densidad_mezcla,
  alpha = alpha_emv,
  nu = nu_emv,
  method = "L-BFGS-B",
  lower = min(datosPerdida$Pérdida), # Límite inferior para la búsqueda
  upper = max(datosPerdida$Pérdida)  # Límite superior para la búsqueda
)

# Usar optim para encontrar la moda de la función de densidad
resultado_moda2 <- optim(
  par = valor_inicial2,
  fn = neg_densidad_mezcla,
  alpha = alpha_emv,
  nu = nu_emv,
  method = "L-BFGS-B",
  lower = min(datosPerdida$Pérdida), # Límite inferior para la búsqueda
  upper = max(datosPerdida$Pérdida)  # Límite superior para la búsqueda
)

# La moda es el punto donde la función de densidad es máxima
moda1 <- resultado_moda$par
moda2 <- resultado_moda2$par
```

Con lo cual los valores de las modas son 57.8813199 y 28.1514919.

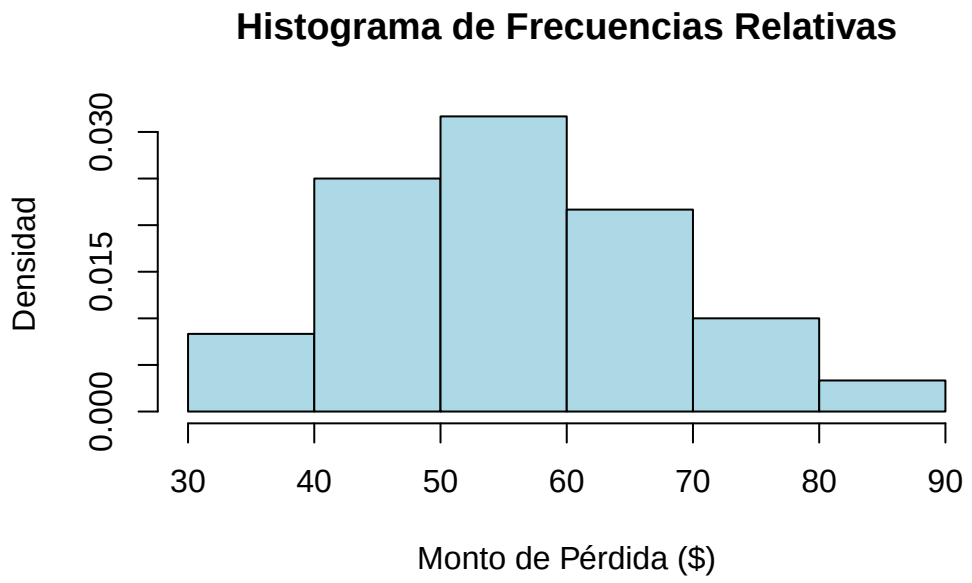
Ejercicio 8

Pruebas de bondad de ajuste. A continuación se muestran los montos (en miles de dólares) de 60 siniestros correspondientes a una póliza de incendio:

a) Construya un histograma de frecuencias relativas con 6 clases de igual longitud entre $30,000$ y $90,000$ dólares. ¿Qué distribuciones paramétricas podrían ajustar a este perfil de siniestros?

[10]

```r
library(readxl)
library(kableExtra)
ruta <- paste0(getwd(), "/bases/Tarea 3.xlsx")
datosPerdida <- read_excel(ruta, sheet = "8", range = "L6:M66")

hist(datosPerdida$Monto,
     breaks = seq(30, 90, length.out = 7), # Define los límites de las clases
     freq = FALSE, # Usa frecuencias relativas en lugar de frecuencias absolutas
     main = "Histograma de Frecuencias Relativas",
     xlab = "Monto de Pérdida ($)",
     ylab = "Densidad",
     col = "lightblue",
     border = "black")
```



b) Si se decide ajustar una Distribución Lognormal con parámetros $\mu \in \mathbb{R}$ y $\sigma^2 > 0$, calcule sus respectivos EMV. Grafique la función de distribución acumulada ajustada junto con la Ojiva. ¿Qué puede concluir? (Sugerencia: Recuerde que estos estimadores fueron deducidos en el Ejercicio E19). [10]

Primero notemos que la función de verosimilitud es de la forma:

$$L(\mu, \sigma \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{x_i \sigma \sqrt{2\pi}} \exp\left(-\frac{(\ln x_i - \mu)^2}{2\sigma^2}\right)$$

y la log-verosimilitud es de la forma:

$$\ln L(\mu, \sigma \mid \underline{X}_n) = -n \ln(\sigma) - \frac{n}{2} \ln(2\pi) - \sum_{i=1}^{n} \ln(x_i) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (\ln x_i - \mu)^2$$

```r
# Optimizamos numéricamente respecto a ambos parámetros

lognormal <- function(x, mu, sigma) {
  dlnorm(x, meanlog = mu, sdlog = sigma)
}

verosimilitud <- function(mu, sigma, datos) {
  log_ver_ind <- log(lognormal(datos, mu, sigma)+.00000001)
  resultado <- sum(log_ver_ind)
  return(resultado)
}

neg_verosimilitud <- function(params, datos) {
  mu <- params[1]
  sigma <- params[2]
  -verosimilitud(mu, sigma, datos)
}

# Suponiendo que tienes un dataframe llamado datosPerdida con una columna 'Pérdida'
datos <- datosPerdida$Monto

# Valores iniciales para alpha y nu
valores_iniciales <- c(mu = log(40), sigma = 10)

# Límites para alpha y nu
# alpha está entre 0 y 1, y nu es mayor que 0
limites <- list(c(.00001, 90), c(.00001, 90))

# Utilizar optim con el método "L-BFGS-B" para encontrar los parámetros óptimos
resultado_optim <- optim(
  par = valores_iniciales,
  fn = neg_verosimilitud,
  datos = datos,
  method = "L-BFGS-B",
  lower = sapply(limites, `[`, 1), # Límites inferiores
  upper = sapply(limites, `[`, 2)  # Límites superiores
```

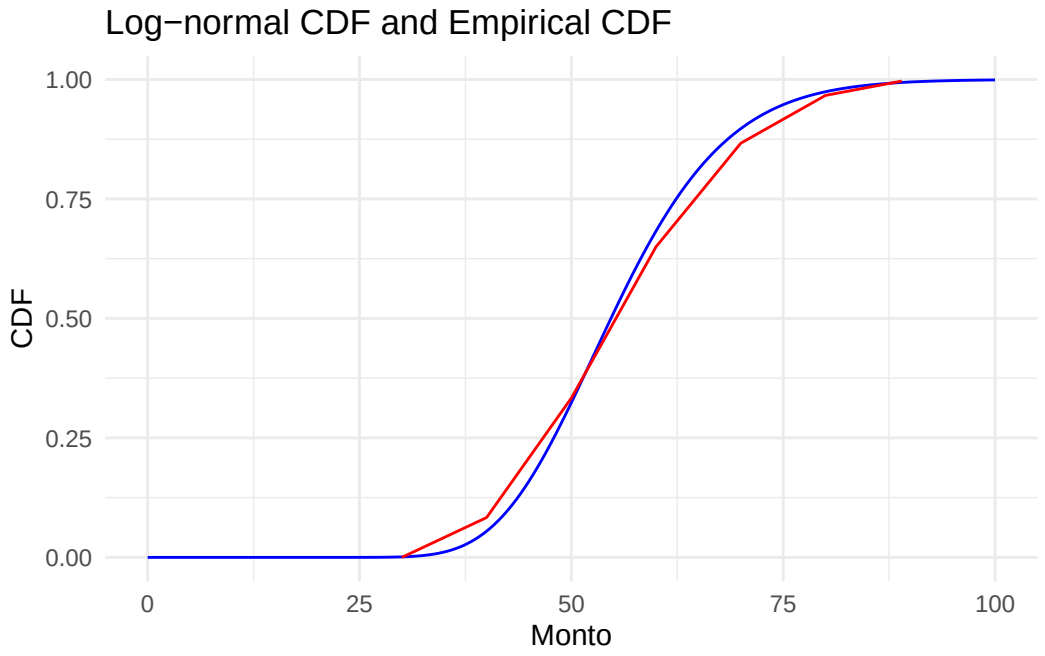
```
)

# Los parámetros optimizados
mu_optim <- resultado_optim$par[1]
sigma_optim <- resultado_optim$par[2]
```

Con lo cual obtenemos que $\hat{\mu} = 4.0013712$ y $\widehat{\sigma^2} = 0.0381647$.

Ahora procederemos a graficar la función de distribución acumulada ajustada junto con la Ojiva.

```
mu_est <- mu_optim
sigma_est <- sigma_optim

# Define una secuencia de valores para los cuales se calculará la ojiva
x_values <- seq(0,100, length.out = 1000)
x_values2 <- seq(30,89, length.out = 1000)
ecdf_datos <- ecdf(datosPerdida$Monto)
# Función de la ojiva

ojiva_lognormal <- function(x){
  particion <- seq(30, 90, length.out = 7)
  for (i in particion){
    if(x <= i & x >= i-10){
      cparticionBefore <- i-10
      cparticionNow <- i
      resultado <-  (cparticionNow -
      x)*ecdf_datos(cparticionBefore)/(cparticionNow-
      cparticionBefore) +
      (x - cparticionBefore)*
      ecdf_datos(cparticionNow)/(cparticionNow- cparticionBefore)
      break
    }
  }
  return(resultado)
}
ojiva_empirica_values <- sapply(x_values2, ojiva_lognormal)

lognormal <- function(x) {
  plnorm(x, meanlog = mu_est, sdlog = sigma_est)
}
```

```r
library(ggplot2)
# Crear el gráfico con ggplot2
ggplot() +
  geom_line(aes(x= x_values, y = lognormal(x_values)), color = "blue") +
  geom_line(aes(x=x_values2, y = ojiva_empirica_values), color = "red") +
  labs(title = 'Log-normal CDF and Empirical CDF', x = 'Monto', y = 'CDF') +
  theme_minimal() +
  theme(legend.position = "bottomright")
```

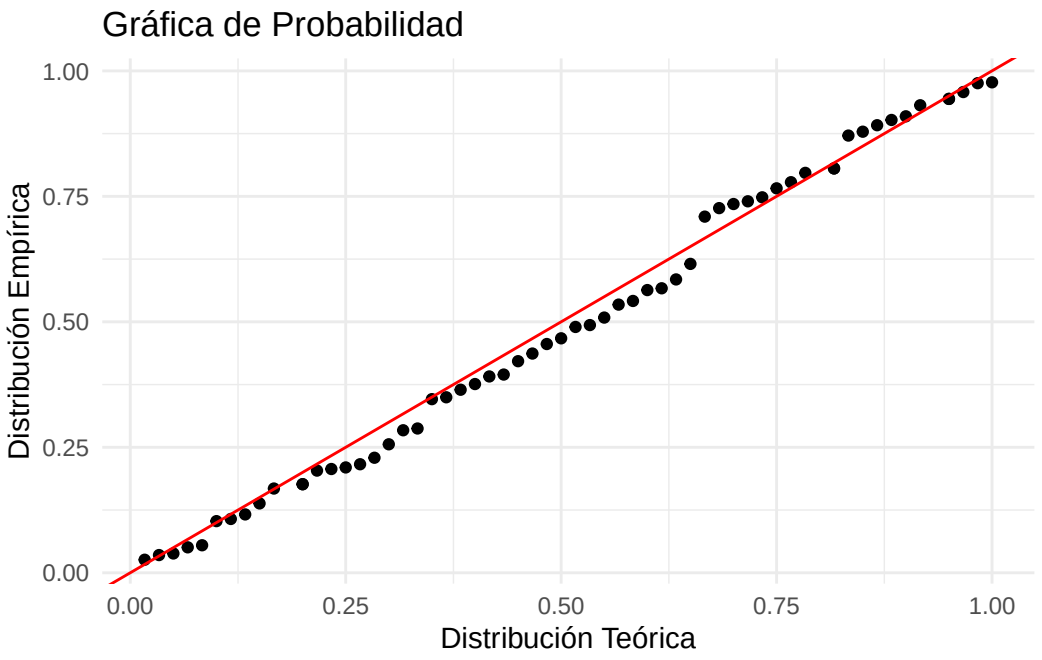


Tenemos que tomar en cuenta que la ojiva es un estimador de la función de distribución acumulada similar al histograma. Y al paracer nuestra distribución ajustada sí está pareciéndose a la distribución de los datos. Solo cabe resaltar que subestima abtes de los 50,000 y sobreestima después de los 50,000.

c) Construya la Gráfica de Probabilidad para el ajuste del inciso b). ¿Qué puede concluir? [10]

```r
# Carga la librería necesaria para funciones adicionales si es necesario
# Por ejemplo, para la distribución t, necesitas la función qt de la librería stats
library(stats)

# Genera tus datos
```

```r
datos <- datosPerdida$Monto

# Distribución empírica
dist_empirica <- ecdf(datos)
datos_empirica <- dist_empirica(datos)
# Distribución teórica
datos_teoricos <- lognormal(datos)

data <- data.frame(datos_empirica, datos_teoricos)

# Crea la gráfica de probabilidad
ggplot(data,aes(x=datos_empirica , y= datos_teoricos)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "red") +
  labs(title = "Gráfica de Probabilidad", x = "Distribución Teórica",
   y = "Distribución Empírica") +
  theme_minimal()
```



Con esta gráfica es mucho más claro y preciso poder decir que la nuestra distribución ajustada se parece a la de los datos muestreados y por tanto no parecen haber grandes desviaciones. Fue un buen ajuste.

d) Aplique la Prueba Kolmogorov-Smirnov para validar el ajuste Lognormal del inciso b).

Considere un tamaño de error tipo I del 5%. [10]

```
# Realizar la prueba de Kolmogorov-Smirnov
ks_result <- ks.test(datosPerdida$Monto, "plnorm", meanlog = mu_est, sdlog = sigma_est)
```

```
Warning in ks.test.default(datosPerdida$Monto, "plnorm", meanlog = mu_est, :
ties should not be present for the Kolmogorov-Smirnov test
```

```
# Imprimir el resultado
print(ks_result)
```

```

    Asymptotic one-sample Kolmogorov-Smirnov test

data:  datosPerdida$Monto
D = 0.059849, p-value = 0.9826
alternative hypothesis: two-sided
```

Con lo cual no se rechaza la hipótesis nula de que la distribución de los datos es lognormal, es decir, el ajuste fue bueno estadísticamente.

e) Aplique la Prueba Ji-Cuadrada para validar el ajuste Lognormal del inciso b). Considere la partición del inciso a) y agregue 2 clases (inicial y final) para abarcar todo el soporte de la Lognormal. Calcule el valor-p para concluir. [10]

```
mis_datos <- datosPerdida$Monto
# Crear una secuencia de intervalos para la tabla de frecuencia
breaks1 <- seq(30, 90, length.out = 7)  # Ajusta el
#número de intervalos según sea necesario
breaks <- c(-Inf, breaks1, Inf)

# Frecuencias observadas: conteo de datos en cada intervalo
observed <- hist(mis_datos, breaks = breaks, plot = FALSE)$counts

# probabilidades
probabilities <- plnorm(breaks, meanlog = mu_est, sdlog = sigma_est)
diff_prob <- diff(probabilities)

# Realizar la prueba chi-cuadrada
pruebachi <- chisq.test(x = observed, p = diff_prob, rescale.p = TRUE)
```

```
Warning in chisq.test(x = observed, p = diff_prob, rescale.p = TRUE):
Chi-squared approximation may be incorrect
```

```r
kparam <- length(diff_prob) # Número de cajitas
qparam <- 2 # parámetros de la distribución ajustada
df_chi <- kparam - qparam - 1 # Grados de libertad
pvalue <- pchisq(pruebachi$statistic, df = df_chi, lower.tail = FALSE)  # Valor-p
```

Por lo que el estadístico de prueba es 2.6457774 y el valor-p es 0.7543982. Es decir, no hay suficiente evidencia para rechazar la hipótesis nula de que la distribución de los datos es lognormal. Otra vez fue un buen ajuste.

Ejercicio 9

Selección de modelos. A continuación se muestra el diagrama de tallo y hojas del monto de las reclamaciones de un seguro que cubre daños ocasionados por fenómenos hidrometeorológicos:

```r
datos <- read_excel(ruta, sheet = "9", range = "M7:N23")
```

a) Suponga que se quiere ajustar una Distribución Gamma con media $\mu = \alpha\theta$.

b) Calcule los EMV de α y θ. (Sugerencia: Recuerde que en el Ejercicio E14 se obtuvieron estos estimadores o utilice EViews para obtenerlos). [5]

Recordemos que la log de la función de máxima verosimilitud está dada por:

$$\ln L(\alpha, \theta) = (\alpha - 1) * \sum_{i=1}^{n} \ln(x_i) - \sum_{i=1}^{n} \frac{x_i}{\theta} - n \ln(\Gamma(\alpha)) - n\alpha \ln(\theta)$$

Y derivando parcialmente con respecto a cada parámetro obtenemos las siguientes ecuaciones:

$$\sum_{i=1}^{n} \ln(x_i) - n\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} - n \ln(\theta) = 0$$

$$\frac{\sum_{i=1}^{n} x_i}{\theta^2} - n\frac{\alpha}{\theta} = 0$$

Con lo cual simplificando obtenemos que

$$\theta = \frac{\bar{x}}{\alpha}$$

$$\sum_{i=1}^{n} \ln(x_i) - n\frac{\Gamma'(\alpha)}{\Gamma(\alpha)} - n\ln(\theta) = 0$$

Es decir, solo hay que utilizar un método numérico para encontrar el valor de α que cumple la segunda ecuación y con ese valor encontrar el valor de θ.

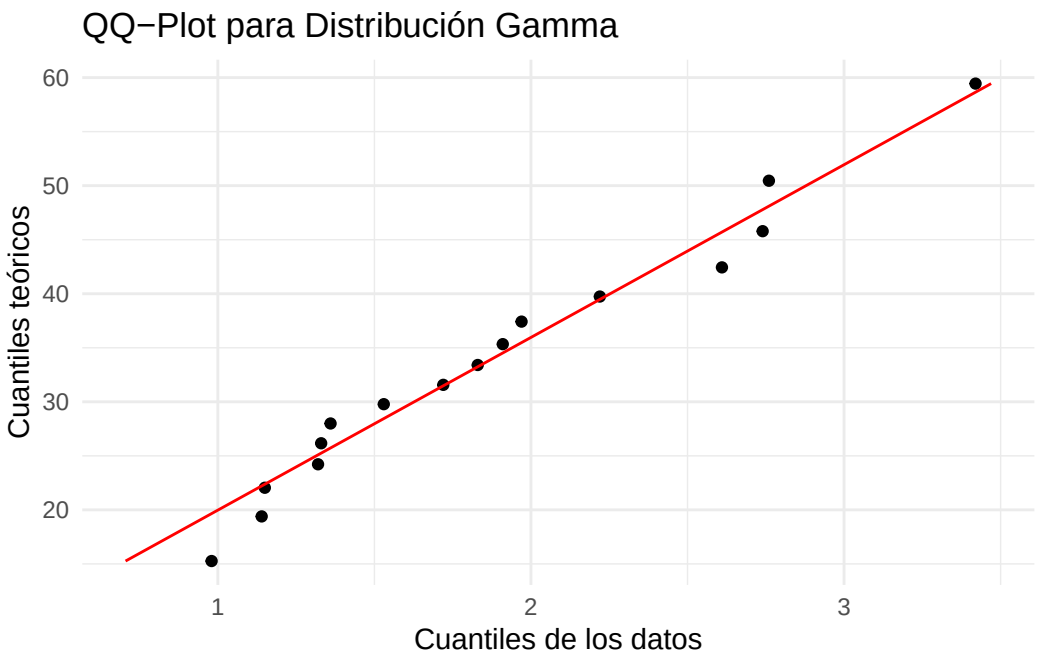
```r
# Definir la función que representa la segunda ecuación
funcion_objetivo <- function(alpha, x) {
  n <- length(x)
  sum(log(x)) - n * digamma(alpha) - n * log(mean(x) / alpha)
}

# Usar uniroot para encontrar el valor de alpha
# Es importante dar un intervalo donde se espera encontrar la raíz (alpha)
resultado <- uniroot(funcion_objetivo, interval = c(0.001, 100), x = datos$Monto)

# El valor de alpha
alpha_estimado <- resultado$root

# Ahora calcular theta usando la relación dada
theta_estimado <- mean(datos$Monto) / alpha_estimado
```

Con lo cual los valores de $\hat{\alpha}$ y $\hat{\theta}$ son 7.9694794 y 0.2351942 respectivamente.

ii) Obtenga el Gráfico Cuantil-Cuantil para el ajuste de la distribución Gamma. ¿Qué puede concluir? (Sugerencia: Utilice EViews). [5]

```r
library(ggplot2)

# Ordena tus datos y calcula los cuantiles empíricos
datos$empirical <- datos$Monto
datos$theoretical <- qgamma(ppoints(length(datos$Monto)), shape = alpha_estimado, rate = t

ggplot(data.frame(x = datos$theoretical, y = datos$empirical), aes(sample = y)) +
  stat_qq(distribution = qgamma, dparams = list(shape = alpha_estimado,
  rate = theta_estimado)) +
  stat_qq_line(distribution = qgamma, dparams = list(shape = alpha_estimado,
  rate = theta_estimado), color = "red") +
```

```
  labs(title = "QQ-Plot para Distribución Gamma",
  x = "Cuantiles teóricos", y = "Cuantiles de los datos") +
  coord_flip()+
  theme_minimal()
```



Con lo cual podemos ver que la distribución Gamma es un buen ajuste para los datos.

iii) Aplique la Prueba de Razón de Verosimilitudes con un nivel de significancia del 5% para determinar si el monto de las reclamaciones corresponde a una Gamma con media $\mu_0 = 2$. (Sugerencia: Note que a pesar de que la distribución Gamma tiene 2 parámetros, la prueba se debe hacer sólo con un parámetro pues $T(\mu) = (\alpha, \theta) = \alpha\theta$, con $T : \mathbb{R}^2 \to \mathbb{R}$. No olvide obtener el EMV restringido considerando el valor de H_0). [15]

```
logverosimilitud <- function(alpha, theta, x) {
  n <- length(x)
  resultado <- (alpha - 1) * sum(log(x)) - sum(x / theta) -
  n * log(gamma(alpha)) - n * alpha * log(theta)
  return(resultado)
}

logverosimilitud_restringida <- function(alpha, x) {
  n <- length(x)
```

```r
  theta <- 2 / alpha
  resultado <- (alpha - 1) * sum(log(x)) - sum(x / theta) -
  n * log(gamma(alpha)) - n * alpha * log(theta)
  return(resultado)
}

# Obtener el EMV restringido

neg_logverosimilitud_restringida <- function(alpha, x) {
  -logverosimilitud_restringida(alpha, x)
}

resultado_optim <- optim(
  par = 1,
  fn = neg_logverosimilitud_restringida,
  x = datos$Monto,
  method = "L-BFGS-B",
  lower = 0.00001, # Límite inferior para alpha
  upper = 100  # Límite superior para alpha
)

alpha_restringido <- resultado_optim$par
theta_restringido <- 2 / alpha_restringido

logverosimilitud_restringida <- logverosimilitud(alpha_restringido,
 theta_restringido, datos$Monto)

logverosimilitud_original <- logverosimilitud(alpha_estimado,
theta_estimado, datos$Monto)

LR_test <- 2 * (logverosimilitud_original -  logverosimilitud_restringida)

valorptest <- 1 - pchisq(LR_test, df = 1)
```

Con lo cual obtuvimos que el valor de la razón de verosimilitudes es 0.517096 y el valor-p es 0.4720831. Por lo que no se rechaza la hipótesis nula de que el monto de las reclamaciones corresponde a una Gamma con media $\mu_0 = 2$.

b) Suponga que se quiere ajustar una distribución Weibull (γ, θ), $\gamma > 0$ y $\theta > 0$.

Primero notemos que la función de verosimilitud está dada por:

$$L(\theta,\gamma) = \frac{\gamma^n}{\theta^{\gamma n}} \left(\Pi_{i=1}^n x_i\right)^{\gamma-1} e^{-\sum_{i=1}^n (\frac{x_i}{\theta})^\gamma}$$

Con lo cual la log-verosimilitud está dada por:

$$\ln L(\theta,\gamma) = n\ln(\gamma) - \gamma n \ln(\theta) + (\gamma-1)\sum_{i=1}^n \ln(x_i) - \sum_{i=1}^n \left(\frac{x_i}{\theta}\right)^\gamma$$

i) Obtenga los EMV de γ y θ. Calcule su log-verosimilitud y grafique la f.d.a. empírica y la f.d.a. ajustada. ¿Qué puede concluir? [10]

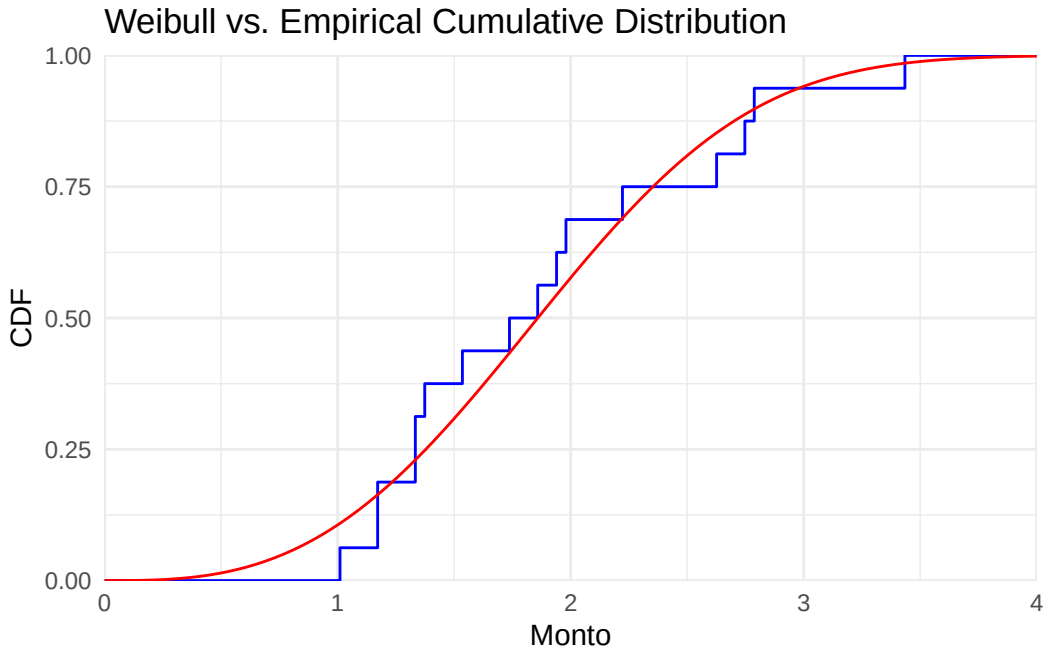
```r
#Tambien se pudo usar de MASS  fitdistr(x, densfun, start, ...)
logveromisimilitud <- function(x, theta, gamma) {
  n <- length(x)
  resultado <- n * log(gamma) - gamma * n * log(theta) +
  (gamma - 1) * sum(log(x)) - sum((x / theta)^gamma)
  return(resultado)
}

neg_logverosimilitud <- function(params, x) {
  theta <- params[1]
  gamma <- params[2]
  -logveromisimilitud(x, theta, gamma)
}

# Valores iniciales para theta y gamma
valores_iniciales <- c(theta = 1, gamma = 1)

# Límites para theta y gamma
limites <- list(c(0.00001, 100), c(0.00001, 100))

# Utilizar optim con el método "L-BFGS-B" para encontrar los parámetros óptimos

resultado_optim <- optim(
  par = valores_iniciales,
  fn = neg_logverosimilitud,
  x = datos$Monto,
  method = "L-BFGS-B",
  lower = sapply(limites, `[`, 1), # Límites inferiores
  upper = sapply(limites, `[`, 2)  # Límites superiores
)
```

```r
# Los parámetros optimizados

theta_estimado <- resultado_optim$par[1]
gamma_estimado <- resultado_optim$par[2]

logver1 <- logveromisimilitud(datos$Monto, theta_estimado, gamma_estimado)
```

Con lo cual los valores de $\hat{\theta}$ y $\hat{\gamma}$ son 2.105692 y 2.940197 respectivamente. Además, el valor de la log-verosimilitud es -16.2594217.

Ahora graficaremos la f.d.a. empírica y la f.d.a. ajustada.

```r
# Crear un nuevo dataframe con valores para las funciones de distribución acumulativa
n <- 100
valores <- seq(0, 4, length.out = n)
df_weibull <- data.frame(
  Monto = valores,
  FittedWeibull = pweibull(valores, shape = gamma_estimado, scale = theta_estimado)
)

# Crear la distribución empírica acumulada (ECDF)
df_empirica <- data.frame(
  Monto = valores,
  Empirical = ecdf(datos$Monto)(valores)
)

# Combinar ambos dataframes
df_combined <- merge(df_weibull, df_empirica, by = "Monto")

# Gráfico con ggplot2
ggplot() +
  geom_step(aes(x = Monto, y = Empirical), data = df_empirica, color = "blue") +
  geom_line(aes(x = Monto, y = FittedWeibull), data = df_weibull, color = "red") +
  labs(title = "Weibull vs. Empirical Cumulative Distribution",
       x = "Monto",
       y = "CDF") +
  theme_minimal() +
  scale_x_continuous(expand = c(0, 0)) +
  scale_y_continuous(expand = c(0, 0))
```



Al parecer fue un buen ajuste pues la f.d.a. ajustada se parece mucho a la f.d.a. empírica. Pero cabe señalar que la f.d.a. ajustada sobreestima poco antes de 1.5 y mayor a 2.3 aproximadamente, y subestima en los otros casos.

ii) Para el ajuste del inciso anterior aplique las pruebas Kolmogorov-Smirnov y Anderson-Darling considerando un nivel de significancia del 5%. (Sugerencia: Utilice EViews). [5]

```
# Realizar la prueba de Kolmogorov-Smirnov
ks_result <- ks.test(datos$Monto, "pweibull",
shape = gamma_estimado, scale = theta_estimado)

print(ks_result)
```

```
    Exact one-sample Kolmogorov-Smirnov test

data:  datos$Monto
D = 0.13339, p-value = 0.9031
alternative hypothesis: two-sided
```

```
# Realizar la prueba de Anderson-Darling
library(ADGofTest)
```

```r
# Define una función de distribución acumulada para la Weibull con tus parámetros estimado
pweibull_with_params <- function(q) {
  pweibull(q, shape = gamma_estimado, scale = theta_estimado)
}

# Realiza la prueba de Anderson-Darling usando tu
#vector de datos y la función de distribución definida
ad_result <- ADGofTest::ad.test(datos$Monto, pweibull_with_params)

# Ver los resultados de la prueba
print(ad_result)
```

```
    Anderson-Darling GoF Test

data:  datos$Monto  and  pweibull_with_params
AD = 0.38855, p-value = 0.8582
alternative hypothesis: NA
```

En las dos pruebas no se rechaza la hipótesis nula de que la distribución de los datos es Weibull. Por lo que fue un buen ajuste.

iii) Suponga que $\gamma = 3$ y que se aplica un deducible de 1.5 millones de dólares. Obtenga en este caso el EMV de θ y calcule su log-verosimilitud. [10]

En este caso solo notemos que ahora la función de verosimilitud está dada por:

$$L(\theta) = \frac{\frac{3^n}{\theta^{3n}} \left(\Pi_{i=1}^n x_i\right)^2 e^{-\sum_{i=1}^n (\frac{x_i}{\theta})^3}}{e^{(\frac{1.5}{\theta})^{3n}}}$$

Con lo cual la log-verosimilitud está dada por:

$$\ln L(\theta) = n\ln(3) - 3n\ln(\theta) + 2\sum_{i=1}^n \ln(x_i) - \sum_{i=1}^n \left(\frac{x_i}{\theta}\right)^3 + n\left(\frac{1.5}{\theta}\right)^3$$

Considerando que ahora n es el número de datos mayores al deducible, los demás los ignoramos.

```r
datosDeducible <- datos[datos$Monto >= 1.5,]

# logVerosimilitud
```

```r
logveromisimilitud <- function(x, theta) {
  n <- length(x)
  resultado <- n * log(3) - 3 * n * log(theta) +
  (3 - 1) * sum(log(x)) - sum((x / theta)^3)+n*(1.5/theta)^3
  return(resultado)
}

neglogverosimilitud <- function(params, x) {
  theta <- params[1]
  -logveromisimilitud(x, theta)
}

# Valores iniciales para theta
valores_iniciales <- 1

# Límites para theta
limites <- list(c(0.00001, 100))

# Utilizar optim con el método "L-BFGS-B" para encontrar los parámetros óptimos

resultado_optim <- optim(
  par = valores_iniciales,
  fn = neglogverosimilitud,
  x = datosDeducible$Monto,
  method = "L-BFGS-B",
  lower = sapply(limites, `[`, 1), # Límites inferiores
  upper = sapply(limites, `[`, 2)  # Límites superiores
)

# Los parámetros optimizados

theta_estimado <- resultado_optim$par[1]
logver2 <- logveromisimilitud(datosDeducible$Monto, theta_estimado)
```

Por lo tanto, el valor de $\hat{\theta}$ es 2.1965479 y el valor de la log-verosimilitud es -6.8070177.

iv) Suponga que $\gamma = 3$ y que no se aplica deducible pero se establece un límite de aseguramiento de 2.5 millones de pesos. Obtenga en este caso el EMV de θ y calcule su log-verosimilitud. [10]

En este caso la función de verosimilitud está dada por:

$$L(\theta) = \frac{3^{m_1}}{\theta^{3m_1}} \left(\Pi_{i=1}^{m_1} x_i\right)^2 e^{-\sum_{i=1}^{m_1}(\frac{x_i}{\theta})^3} (e^{-\sum_{i=1}^{m_2}(\frac{2.5}{\theta})^3})$$

donde m_1 es el número de datos menores al límite de aseguramiento y m_2 es el número de datos mayores al límite de aseguramiento.

Y por lo tanto la log-verosimilitud está dada por:

$$\ln L(\theta) = m_1 \ln(3) - 3m_1 \ln(\theta) + 2\sum_{i=1}^{m_1} \ln(x_i) - \sum_{i=1}^{m_1} \left(\frac{x_i}{\theta}\right)^3 - \sum_{i=1}^{m_2} \left(\frac{2.5}{\theta}\right)^3$$

```
logveromisimilitud <- function(x, theta) {
  n <- length(x)
  xind <- x[x < 2.5]
  xout <- x[x >= 2.5]
  m1 <- length(xind)
  m2 <- length(xout)
  resultado <- m1 * log(3) - 3 * m1 * log(theta) +
  (3 - 1) * sum(log(xind)) - sum((xind / theta)^3)-m2*(2.5 / theta)^3
  return(resultado)
}

neglogverosimilitud <- function(params, x) {
  theta <- params[1]
  -logveromisimilitud(x, theta)
}

# Valores iniciales para theta
valores_iniciales <- 1

# Límites para theta
limites <- list(c(0.00001, 100))

# Utilizar optim con el método "L-BFGS-B" para encontrar los parámetros óptimos

resultado_optim <- optim(
  par = valores_iniciales,
  fn = neglogverosimilitud,
  x = datos$Monto,
  method = "L-BFGS-B",
  lower = sapply(limites, `[`, 1), # Límites inferiores
```

```
  upper = sapply(limites, `[`, 2)  # Límites superiores
)

# Los parámetros optimizados

theta_estimado <- resultado_optim$par[1]
logver3 <- logveromisimilitud(datos$Monto, theta_estimado)
```

Con lo cual el valor de $\hat{\theta}$ es 2.1176923 y el valor de la log-verosimilitud es -16.1975445.

v) Calcule los Criterios de Información de Akaike, Schwartz y Hannan-Quinn para los modelos ajustados en los subincisos i), iii) y iv). ¿Cuál modelo se prefiere? Justifique brevemente su respuesta. [5]

```
# Criterios
akaike_criterio <- function(logverosimilitud, k, n) {
  resultado <- (-2 * logverosimilitud + 2 * k)/n
  return(resultado)
}

schwartz_criterio <- function(logverosimilitud, k, n) {
  resultado <- (-2 * logverosimilitud + 2 * log(n))/n
  return(resultado)
}

hannan_quinn_criterio <- function(logverosimilitud, k, n) {
  resultado <- (-2 * logverosimilitud + 2 * k * log(log(n)))/n
  return(resultado)
}
```

	(i)	(ii)	(iii)
n	16	10	16
Log-verosimilitud	-16.2594217	-6.8070177	-16.1975445
q	2	1	1
Criterios de información			
Akaike	2.2824277	1.5614035	2.1496931
Schwartz	2.3790013	1.8219206	2.3712666
Hannan-Quinn	2.2873731	1.52821	2.1521657

Notamos que el modelo que tiene menor valor en los 3 criterios de información es el modelo

del inciso iii). Por lo que es el modelo que más se preferirá y el que mejor ajustó a los datos siguiendo cierta parsimonía con el número de parámetros que se estimaron. Es decir, el modelo con límite de aseguramiento (sin tomar en cuenta deducible) es mejor para poder ajustar los datos a una Weibull.

Ejercicio 11: Aproximación Binomial Negativa por Poisson

Suponga que

$$Y \sim \text{Po}(\lambda), \quad \lambda > 0;$$

$$N_i \sim \text{iid Geo}(p), \quad 0 < p < 1;$$

y que

$$N = N_1 + N_2 + ... + N_r, \quad r \in \mathbb{Z}^+.$$

d) Para $N \sim \text{BN}(50, 0.2)$ se quieren obtener $P(N = 0)$ y $P(N - E[N] \leq \sigma)$.

i) Calcule exactamente ambas probabilidades. [5]

$$P(N = 0)$$

$$P(N - E[N] \leq \sigma)$$

```
# Establecer los parámetros para la primera
size <- 50    # Número deseado de éxitos
prob <- .2    # Probabilidad de éxito en cada ensayo
x <- 0

# Calcular la probabilidad
probabilidad1 <- dnbinom(x, size, prob)

#-----------------------------
# Parámetros de la distribución binomial negativa
size <- 50  # número de éxitos deseado
prob <- 0.2 # probabilidad de éxito en cada ensayo

# Calcular la media y la desviación estándar
media <- size* (1 - prob) / prob
```

```r
varianza <- size * (1 - prob) / (prob^2)
desviacion_std <- sqrt(varianza)

# Calcular la probabilidad de que N esté dentro de una desviación estándar de la media
limite_superior <- pnbinom((desviacion_std+media), size = size, prob = prob)

probabilidad2 <- limite_superior
```

Notemos que las probabilidades son:

$$P(N = 0) = 1.1258999 \times 10^{-35}$$

$$P(N - E[N] \leq \sigma) = 0.8412284$$

ii) Aproxime ambas probabilidades mediante la distribución Poisson. ¿Cuál es el error de la aproximación? [5]

```r
  # Calcular la probabilidad de que N esté dentro de una desviación estándar de la media
lambda <-  size* (1 - prob) / prob
media <-  lambda
varianza <- lambda
desviacion_std <- sqrt(varianza)

limite_superior <- ppois((desviacion_std+media), lambda= lambda)

probabilidad3 <- limite_superior

# calculamos proba de n=0

probabilidad4 <- dpois(0, lambda)

error1 <- abs(probabilidad4-probabilidad1)
error2 <- abs(probabilidad3-probabilidad2)
```

Con lo cual obtenemos lo siguiente

$$P(N = 0) = 1.3838965 \times 10^{-87}$$

$$P(N - E[N] \leq \sigma) = 0.8473017$$

Y el error de aproximación a cada son los siguientes:

$$\text{error}(P(N = 0)) = 1.1258999 \times 10^{-35}$$

$$\text{error}(P(N - E[N] \leq \sigma)) = 0.0060733$$

iii) Aproxime ambas probabilidades mediante el Teorema Central del Límite aplicando el Ajuste de Yate. ¿Cuál es el error de aproximación? ¿Qué puede concluir? [10]

En este caso contruimos $N = \sum_{i=1}^{m} N_i$ donde $N_i \sim \textbf{Geo}(p)$ con lo cual por el TCL tenemos:

$$\frac{\sum_{i=1}^{m} N_i - m\frac{1-p}{p}}{\sqrt{m\frac{1-p}{p^2}}} \dot{\sim} N(0,1)$$

y ahora sí podemos aproximar con Normal y con el ajuste de Yate.

```
media <- size* (1 - prob) / prob

varianza <- size * (1 - prob) / (prob^2)

desviacion_std <- sqrt(varianza)

# Calcular la probabilidad de que N esté dentro de una desviación estándar de la media

limite_superior <- pnorm((desviacion_std+media+.5), mean = media, sd = desviacion_std)

probabilidad5 <- limite_superior

# calculamos proba de n=0

probabilidad6 <- pnorm(0+.5, mean = media, sd = desviacion_std)-
pnorm(0-.5, mean = media, sd = desviacion_std)
```

Con lo cual obtenemos lo siguiente

$$P(N = 0) \approx P(N \leq 0 + .5) - P(N \leq 0 - .5) = 2.6045092 \times 10^{-11}$$

$$P(N - E[N] \leq \sigma) \approx P(N \leq \sigma + E[N] + .5)$$

$$= 0.8451404$$

donde $N \dot{\sim} N(E[N], Var(N))$

Y los errores de aproximación fueron los siguientes:

$$\text{error}(P(N = 0)) = 2.6045092 \times 10^{-11}$$

$$\text{error}(P(N - E[N] \leq \sigma)) = 0.003912$$

Con lo cual notamos que la aproximación con el TCL y el ajuste de Yate fue mucho mejor que la aproximación con la distribución Poisson, al calcular la probabilidad de que $|N - E[N]| \leq \sigma$, sin embargo, es peor al calcular la probabilidad de que $N = 0$. Esto se debe a la naturaleza de las distribuciones involucradas, ya que cuando se refiera a calcular intervalos es mejor la aproximación de una distribución continua mientras que para calcular probabilidades puntuales es mejor la aproximación de una distribución discreta.

Ejercicio 12 Clase (a, b, 0) vs. Clase (a, b, 1).

Suponga que $N \sim \text{Bin}(m, q)$, con $m \in \mathbb{Z}^+$ y $0 < q < 1$.

a) Demuestre que N pertenece a la Clase $(a, b, 0)$ y especifique los valores de los parámetros a y b, así como de $P(N = 0)$. [10]

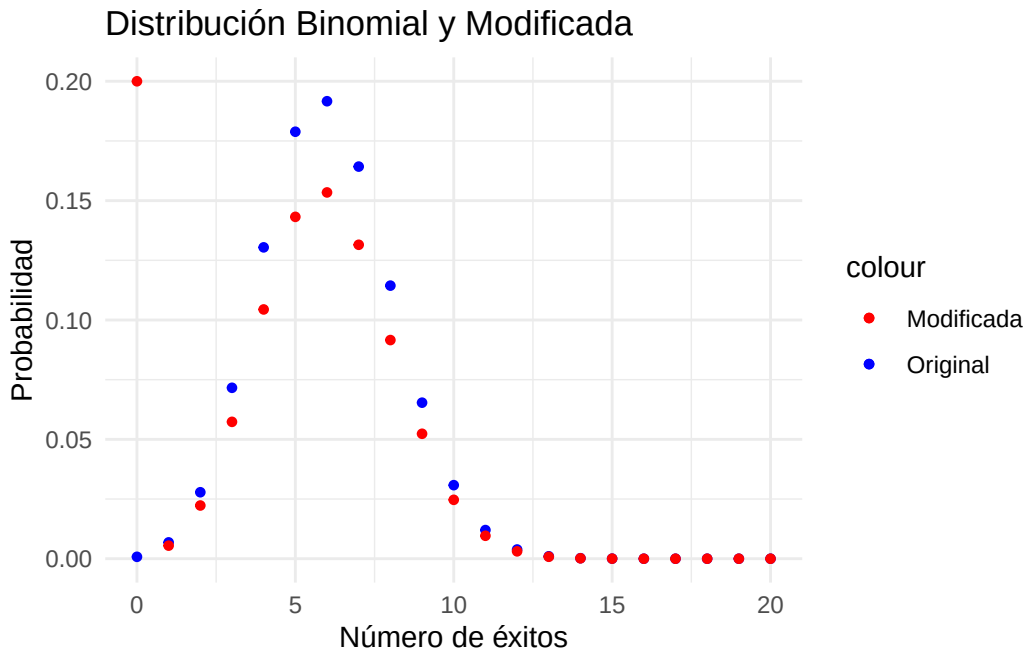
b) Obtenga la función de masa de probabilidad de la distribución Binomial truncada en cero y verifique analíticamente que pertenece a la Clase $(a, b, 1)$. [10]

c) Si $m = 20$ y $q = 0.3$, grafique la función de masa de probabilidad de N junto con la función de masa de probabilidad de su respectiva distribución Binomial modificada en cero considerando $P(N = 0) = 0.2$. Verifique numéricamente que la Binomial modificada en cero pertenece a la Clase $(a, b, 1)$. [10]

Primero notemos que estas son las gráficas de las distribuciones binomiales originales y modificadas en cero.

```
library(ggplot2)

# Parámetros de la distribución binomial
```

```r
n <- 20
p <- 0.3

# Probabilidad modificada en cero
p0_mod <- 0.2

# Vector de éxitos
x <- 0:n

# Función de probabilidad de masa de la distribución binomial original
dbinom_original <- dbinom(x, size = n, prob = p)

# Ajustar la distribución binomial modificada
# Primero, calcula la probabilidad de 0 éxitos de la distribución binomial original
p0_original <- dbinom_original[1]

# Calcula el factor de ajuste
adjust_factor <- (1-p0_mod) / (1 - p0_original)

# Aplica el factor de ajuste a las probabilidades de 1 a n éxitos
dbinom_modified <- dbinom_original
dbinom_modified[-1] <- dbinom_original[-1] * (adjust_factor)

# Asegúrate de que la suma de las probabilidades sea 1
dbinom_modified[1] <- p0_mod

# Crea un dataframe para graficar
df <- data.frame(x, dbinom_original, dbinom_modified)

# Graficar usando ggplot2
ggplot(df, aes(x = x)) +
  geom_point(aes(y = dbinom_original, colour = "Original"), size = 1.2) +
  geom_point(aes(y = dbinom_modified, colour = "Modificada"), size = 1.2) +
  labs(x = "Número de éxitos", y = "Probabilidad") +
  scale_colour_manual(values = c("Original" = "blue", "Modificada" = "red")) +
  ggtitle("Distribución Binomial y Modificada") +
  theme_minimal()
```



Ahora notemos que la binomial modificada pertenece a la clase $(a, b, 1)$ pues si resolvemos el sistema de ecuaciones formado por la relación de recurrencia de las primeras probas tenemos:

$$\frac{p_2}{p_1} = a + \frac{b}{2}$$

$$\frac{p_3}{p_2} = a + \frac{b}{3}$$

```
# Resolvemos el sistema de ecuaciones
# Definir la matriz de coeficientes
A <- matrix(c(1, .5, 1, 1/3), nrow = 2, byrow = TRUE)

# Definir el vector de constantes
B <- c(dbinom_modified[3]/dbinom_modified[2], dbinom_modified[4]/dbinom_modified[3])

# Resolver para X
X <- solve(A, B)
a=X[1]
b=X[2]
```

Con lo cual tenemos que $a = -0.4285714$ y $b = 9$. y

Y solo verificamos que a partir de la relación de recurrencia obtengamos las probabilidades exactas de la distribución binomial modificada.

```
# Definir la función de probabilidad de masa de la distribución binomial modificada

binom_modified_num <- c(dbinom_modified[1], dbinom_modified[2],1:19)

for (i in 1:19){
  binom_modified_num[i+2] <- binom_modified_num[i+1]*(a+b/(i+1))
}

binom_modified_num-dbinom_modified
```

```
 [1] 0.000000e+00 0.000000e+00 0.000000e+00 6.938894e-18 2.498002e-16
 [6] 8.881784e-16 1.804112e-15 2.442491e-15 2.470246e-15 2.060851e-15
[11] 1.332268e-15 6.904199e-16 2.896988e-16 9.963818e-17 2.718637e-17
[16] 5.963112e-18 1.024910e-18 1.344665e-19 1.269557e-20 7.868556e-22
[21] 2.716125e-23
```

Y como podemos ver la diferencia entre la lista obtenida y la lista de las probabilidades de la binomial modificada es cero o muy cercana a cero numéricamente, por lo que la binomial modificada pertenece a la clase $(a, b, 1)$.