Clase Modelado 1

2024-05-24

Supuestos, bases y datos

Modelos

```
# Corremos librerías
library(dplyr)
library(ggplot2)
library(purrr)
library(MASS)
library(fitdistrplus)
library(actuar)

# Datos
# Dataframe con la severidad de los siniestros
sevDatos <- data.frame(
  Rango = c("0-1000", "1000-3000", "3000-5000", "5000-10000", "+10000"),
  Frecuencia = c(79, 37, 4, 6, 9)
)

sevDatos
```

```
      Rango Frecuencia
1     0-1000         79
2  1000-3000         37
3  3000-5000          4
4 5000-10000          6
5    +10000          9
```

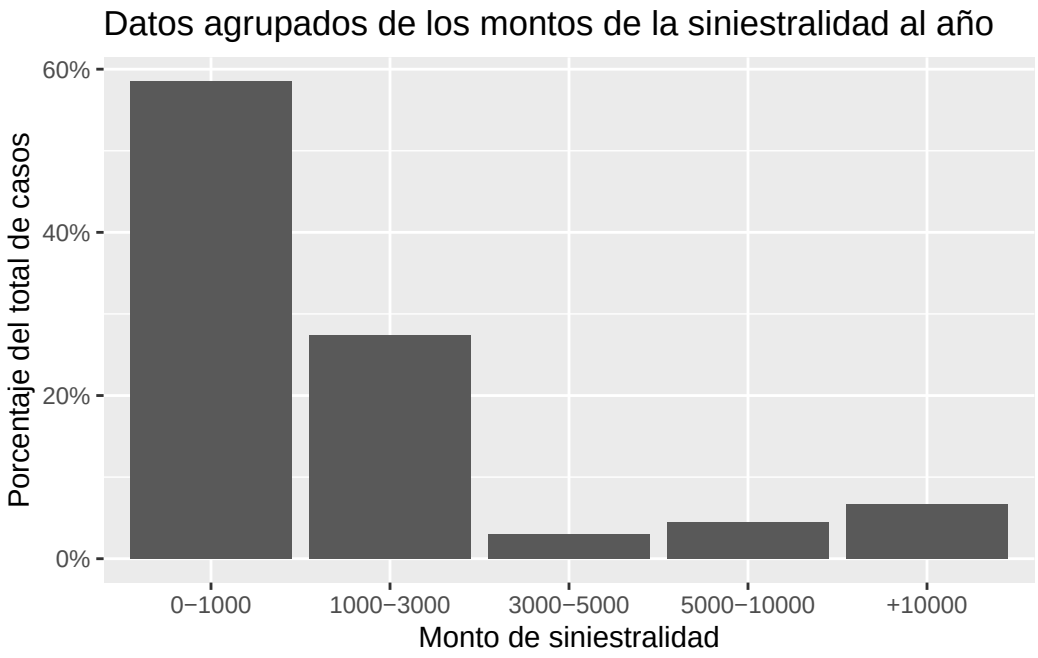
```r
# Dataframe con la frecuencia de los siniestros
frecDatos <- data.frame(
  Rango = c("0--1", "2--3", "3--4"),
  Completa = c(56, 12, 2)
)

frecDatos
```

```
  Rango Completa
1  0--1       56
2  2--3       12
3  3--4        2
```

Análisis exploratorio de datos

Siniestralidad

```r
# Histograma de la severidad

# Calcular los porcentajes
sevDatos$Porcentaje <- sevDatos$Frecuencia / sum(sevDatos$Frecuencia) * 100

# Definir los niveles deseados en el orden correcto
niveles_deseados <- c("0-1000", "1000-3000", "3000-5000", "5000-10000", "+10000")

# Convertir Rango a factor con el orden deseado
sevDatos$Rango <- factor(sevDatos$Rango, levels = niveles_deseados)

# Crear la gráfica con ggplot2
library(ggplot2)
(plot1 <- ggplot(sevDatos, aes(x = Rango, y = Porcentaje)) +
  geom_bar(stat = "identity") +
  labs(title = "Datos agrupados de los montos de la siniestralidad al año",
       x = "Monto de siniestralidad",
       y = "Porcentaje del total de casos") +
  scale_y_continuous(labels = scales::percent_format(scale = 1)))
```



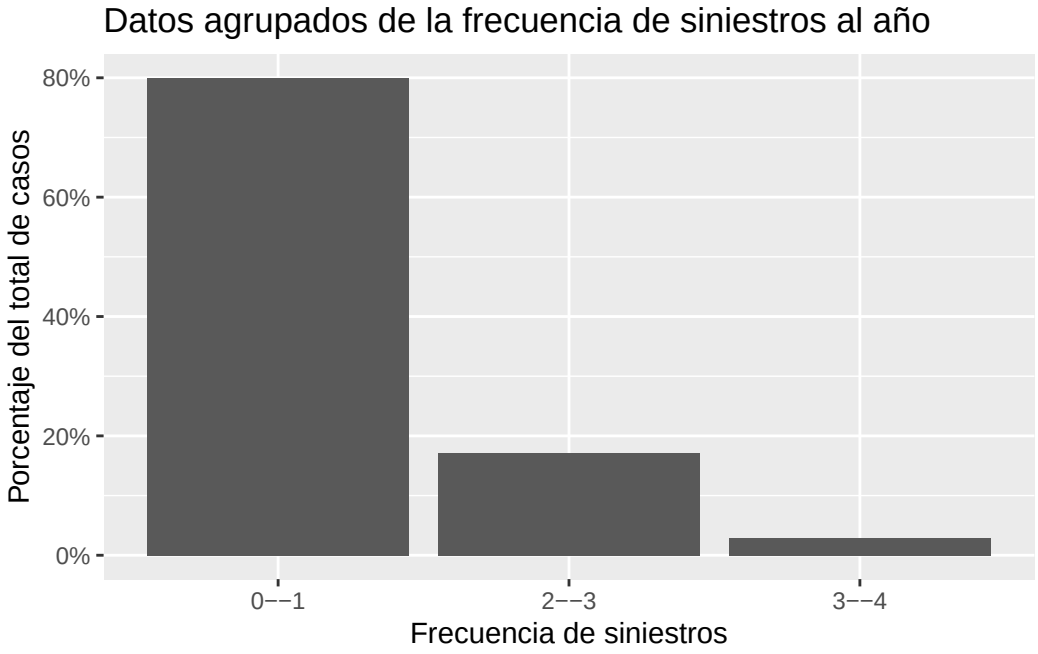

```
# Guardar el gráfico en un archivo JPG
ggsave(filename = "histograma_severidad_porc.jpg", plot = plot1, device = "jpeg")
```

Con lo cual notamos que para la siniestralidad de estos datos, los modelos que convienen deben de ser colas pesadas o medio pesadas, dado que se ve un ligero incremento final en la siniesralidad.

Frecuencia

```
# Histograma de la frecuencia

# Calcular los porcentajes
frecDatos$Porcentaje <- frecDatos$Completa / sum(frecDatos$Completa) * 100

# Definir los niveles deseados en el orden correcto

niveles_deseados <- c("0--1", "2--3", "3--4")

# Convertir Rango a factor con el orden deseado

frecDatos$Rango <- factor(frecDatos$Rango, levels = niveles_deseados)
```

```r
# Crear la gráfica con ggplot2

(plot2 <- ggplot(frecDatos, aes(x = Rango, y = Porcentaje)) +
  geom_bar(stat = "identity") +
  labs(title = "Datos agrupados de la frecuencia de siniestros al año",
       x = "Frecuencia de siniestros",
       y = "Porcentaje del total de casos") +
  scale_y_continuous(labels = scales::percent_format(scale = 1)))
```




```r
# Guardar el gráfico en un archivo JPG

ggsave(filename = "histograma_frecuencia_porc.jpg", plot = plot2, device = "jpeg")
```

```
Saving 5.5 x 3.5 in image
```

Dada la forma de los datos de frecuencia, los modelos que mejor pueden convenir son los modelos de Poisson o Binomial Negativa de la clase(a,b,0) para mantenerlo parsiarmónico.

Ajuste modelos de frecuencia

```
library(stats4)
# Definir los límites de los intervalos
intervalos <- list(c(-1, 1), c(1, 3), c(3,4))

# Crear una función de log-verosimilitud para la distribución de Poisson
logverosimilitud_poisson <- function(lambda) {
  result <- -sum(sapply(1:nrow(frecDatos), function(i) {
    frecDatos$Completa[i] * log(ppois(intervalos[[i]][2], lambda)-ppois(intervalos[[i]][1]
  }))
  return(result)
}

# Crear una función de log-verosimilitud para la distribución binomial negativa
logverosimilitud_negbin <- function(par) {
  size <- par[1]
  mu <- par[2]
  result <- -sum(sapply(1:nrow(frecDatos), function(i) {
    frecDatos$Completa[i] *log(pnbinom(intervalos[[i]][2], size = size, mu = mu)-pnbinom(i
  }))
  return(result)
}


# Ajuste a la distribución de Poisson
ajuste_poisson <- optimize(interval = c(.2,10), f = logverosimilitud_poisson)
(lambda_poisson <- ajuste_poisson$minimum)
```

```
[1] 0.8671189
```

```
# Ajuste a la distribución binomial negativa
ajuste_negbin <- optim(par = c(1, 1), fn = logverosimilitud_negbin)
size_negbin <- ajuste_negbin$par[1]
(mu_negbin <- ajuste_negbin$par[2])
```

```
[1] 0.8158993
```

```r
# AIC()

# AIC para Poisson
(AIC_poisson <- 2 * logverosimilitud_poisson(lambda_poisson) + 2)
```

```
[1] 85.84351
```

```r
# AIC para binomial negativa
(AIC_negbin <- 2 * logverosimilitud_negbin(c(size_negbin, mu_negbin)) + 4)
```

```
[1] 87.11088
```

```r
# Prueba de chi-cuadrado
observados <- frecDatos$Completa

# Esperados para Poisson
esperados_poisson <- sapply(1:nrow(frecDatos), function(i) {
  (ppois(intervalos[[i]][2], lambda_poisson)-ppois(intervalos[[i]][1], lambda_poisson))*su
})


# Esperados para binomial negativa
esperados_negbin <- sapply(1:nrow(frecDatos), function(i) {
  (pnbinom(intervalos[[i]][2], size = size_negbin, mu = mu_negbin)-pnbinom(intervalos[[i]]
})


# Prueba de bonda de ajuste

# Chi-cuadrado para Poisson
chisq_poisson <- sum((observados - esperados_poisson)^2 / esperados_poisson)
p_val_poisson <- pchisq(chisq_poisson, df = nrow(frecDatos) - 1, lower.tail = FALSE)

# Chi-cuadrado para binomial negativa
chisq_negbin <- sum((observados - esperados_negbin)^2 / esperados_negbin)
p_val_negbin <- pchisq(chisq_negbin, df = nrow(frecDatos)- 2, lower.tail = FALSE)

# Resultados de la prueba de chi-cuadrado
cat("Chi-cuadrado para Poisson:", chisq_poisson, "p-value:", p_val_poisson, "\n")
```

```
Chi-cuadrado para Poisson: 2.843979 p-value: 0.2412336
```

```r
cat("Chi-cuadrado para Binomial Negativa:", chisq_negbin, "p-value:", p_val_negbin, "\n")
```

```
Chi-cuadrado para Binomial Negativa: 0.8851139 p-value: 0.3468048
```

Notamos que al ajustar, se valida la bondad de ajuste de los modelos de Poisson y Binomial Negativa; sin embargo, el modelo de Poisson es el que tiene un mejor AIC, parámetro que usaremos para elegir el modelo, dado que nos indica que realiza un mejor ajuste a los datos.

Por último, solo mostraremos los resultados del ajuste gráficamente.

```r
library(ggplot2)

# Crear un data frame con todos los datos
frecDatosComp <- data.frame(
  Rango = factor(frecDatos$Rango),
  Observados = observados,
  Esperados_Poisson = esperados_poisson,
  Esperados_NegBin = esperados_negbin
)

# Convertir los datos a formato largo para ggplot2
frecDatos_long <- reshape2::melt(frecDatosComp, id.vars = "Rango",
                                 variable.name = "Tipo", value.name = "Frecuencia")

# Crear la gráfica de barras
plot3 <- ggplot(frecDatos_long, aes(x = Rango, y = Frecuencia, fill = Tipo)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Comparación de los datos observados y esperados de la frecuencia por model
       x = "Rango",
       y = "Frecuencia") +
  scale_fill_manual(values = c("Observados" = "blue",
                               "Esperados_Poisson" = "red",
                               "Esperados_NegBin" = "green"))

# Guardar el gráfico en un archivo JPG
ggsave(filename = "ajuste_frecuencia.jpg", plot = plot3, device = "jpeg")
```

Ajuste modelos de siniestralidad

Procedimos a ajustar los modelos weibull, gamma, lognormal y pareto a los datos de siniestralidad. Y con los cuales obtuvimos la siguiente gráfica para comparar el ajuste, además de que realizamos una prueba Chi-cuadrado para validar la bondad de ajuste de los modelos.

```
# Ajuste de los datos
library(fitdistrplus)
library(ggplot2)
library(readxl)
# Corregir los datos

set.seed(191654)

mydata <- read_excel("instrumentos.xlsx", sheet = "data")

# Definir una función para calcular el valor intermedio
calcular_intermedio <- function(rango) {
  if (grepl("más de", rango)) {
    lower <- as.numeric(gsub("[^0-9]", "", rango))
    upper <- NA
    intermedio <- lower * 1.5  # Asumimos un incremento del 50% como valor intermedio
  } else {
    nums <- as.numeric(unlist(regmatches(rango, gregexpr("[0-9]+", rango))))
    lower <- nums[1]
    upper <- nums[2]
    intermedio <- mean(c(lower, upper))
  }
  return(intermedio)
}
# Aplicar la función a los datos
mydata$Intermedio <- sapply(mydata$"¿Cuánto cuesta tu instrumento?", calcular_intermedio)

# Definir los límites de los rangos
limites <- list(c(0,1000) ,c(1000, 3000), c(3000, 5000), c(5000, 10000), c(10000, Inf))

# Función de log-verosimilitud para la distribución Weibull
logverosimilitud_weibull <- function(par) {
  shape <- par[1]
  scale <- par[2]
  result <- -sum(sapply(1:nrow(sevDatos), function(i) {
    sevDatos$Frecuencia[i] * log(pweibull(limites[[i]][2], shape, scale) - pweibull(limite
```

```
   }))#+sum(sevDatos$Frecuencia)*log(pweibull(deducible, shape, scale, lower.tail = FALSE)
  return(result)
} # empezar con c(1, 1000)


# Función de log-verosimilitud para la distribución Gamma
logverosimilitud_gamma <- function(par) {
  shape <- par[1]
  rate <- par[2]
  result <- -sum(sapply(1:nrow(sevDatos), function(i) {
    sevDatos$Frecuencia[i] * log(pgamma(limites[[i]][2], shape, scale = rate) - pgamma(lim
  }))#+sum(sevDatos$Frecuencia)*log(pgamma(deducible, shape, scale = rate, lower.tail = FA
  return(result)
}

# Función de log-verosimilitud para la distribución Lognormal
logverosimilitud_lognormal <- function(par) {
  meanlog <- par[1]
  sdlog <- par[2]
  result <- -sum(sapply(1:nrow(sevDatos), function(i) {
    sevDatos$Frecuencia[i] * log(plnorm(limites[[i]][2], meanlog, sdlog) - plnorm(limites[
  }))#+ sum(sevDatos$Frecuencia)*log(plnorm(log(deducible), meanlog, sdlog, lower.tail = F
  return(result)
}

# Función de log-verosimilitud para la distribución Pareto
logverosimilitud_pareto <- function(par) {
  shape <- par[1]
  scale <- par[2]
  result <- -sum(sapply(1:nrow(sevDatos), function(i) {
    sevDatos$Frecuencia[i] * log(ppareto(limites[[i]][2], shape, scale) - ppareto(limites[
  }))#+sum(sevDatos$Frecuencia)*log(ppareto(deducible, shape, scale, lower.tail = FALSE))
  return(result)
}
# Ajustar las distribuciones utilizando optimización
ajustar_distribucion <- function(logverosimilitud, start_params) {
  fit <- optim(start_params, logverosimilitud)
  return(fit)
}

# Ajustar las distribuciones
```

```r
params_weibull <- ajustar_distribucion(logverosimilitud_weibull, start_params = c(1, 1000)
params_gamma <- ajustar_distribucion(logverosimilitud_gamma, start_params = c(10, 1000))
params_lognormal <- ajustar_distribucion(logverosimilitud_lognormal, start_params = c(1, 1
params_pareto <- ajustar_distribucion(logverosimilitud_pareto, start_params = c(10, 1000))

# Calcular AIC
(AIC_weibull <- 2 * logverosimilitud_weibull(params_weibull$par) + 4)
```

```
[1] 309.576
```

```r
(AIC_gamma <- 2 * logverosimilitud_gamma(params_gamma$par) + 4)
```

```
[1] 313.6526
```

```r
(AIC_lognormal <- 2 * logverosimilitud_lognormal(params_lognormal$par) + 4)
```

```
[1] 305.5759
```

```r
(AIC_pareto <- 2 * logverosimilitud_pareto(params_pareto$par) + 4)
```

```
[1] 303.887
```

```r
# Prueba de chi-cuadrada

# Calcular las frecuencias esperadas para cada distribución
calcular_frecuencias_esperadas <- function(distribucion, params, limites, total_observacio
  p <- sapply(1:length(limites), function(i) {
    if (distribucion == "weibull") {
      return((pweibull(limites[[i]][2], params$par[1], params$par[2]) -
                pweibull(limites[[i]][1], params$par[1], params$par[2])))
    } else if (distribucion == "gamma") {
      return((pgamma(limites[[i]][2], params$par[1], rate = params$par[2]) -
                pgamma(limites[[i]][1], params$par[1], rate = params$par[2])))
    } else if (distribucion == "lognormal") {
      return((plnorm(log(limites[[i]][2]), params$par[1], params$par[2]) -
                plnorm(log(limites[[i]][1]), params$par[1], params$par[2])))
```

```
    } else if (distribucion == "pareto") {
      return((ppareto(limites[[i]][2], params$par[1], params$par[2]) -
                ppareto(limites[[i]][1], params$par[1], params$par[2])))
    } else {
      stop("Distribución no soportada")
    }
  })
  return(p * total_observaciones)
}


(total_observaciones <- sum(sevDatos$Frecuencia))
```

```
[1] 135
```

```
expected_weibull <- calcular_frecuencias_esperadas("weibull", params_weibull, limites, tot
expected_gamma <- calcular_frecuencias_esperadas("gamma", params_gamma, limites, total_obs
expected_lognormal <- calcular_frecuencias_esperadas("lognormal", params_lognormal, limite
expected_pareto <- calcular_frecuencias_esperadas("pareto", params_pareto, limites, total_

# Realizar la prueba chi cuadrado para cada distribución
observed <- sevDatos$Frecuencia

(chisq_test_weibull <- chisq.test(x = observed, p = expected_weibull / sum(expected_weibul
```

```
    Chi-squared test for given probabilities with simulated p-value (based
    on 2000 replicates)

data:  observed
X-squared = 10.254, df = NA, p-value = 0.05097
```

```
(chisq_test_gamma <- chisq.test(x = observed, p = expected_gamma / sum(expected_gamma), re
```

```
    Chi-squared test for given probabilities with simulated p-value (based
    on 2000 replicates)

data:  observed
X-squared = Inf, df = NA, p-value = NA
```

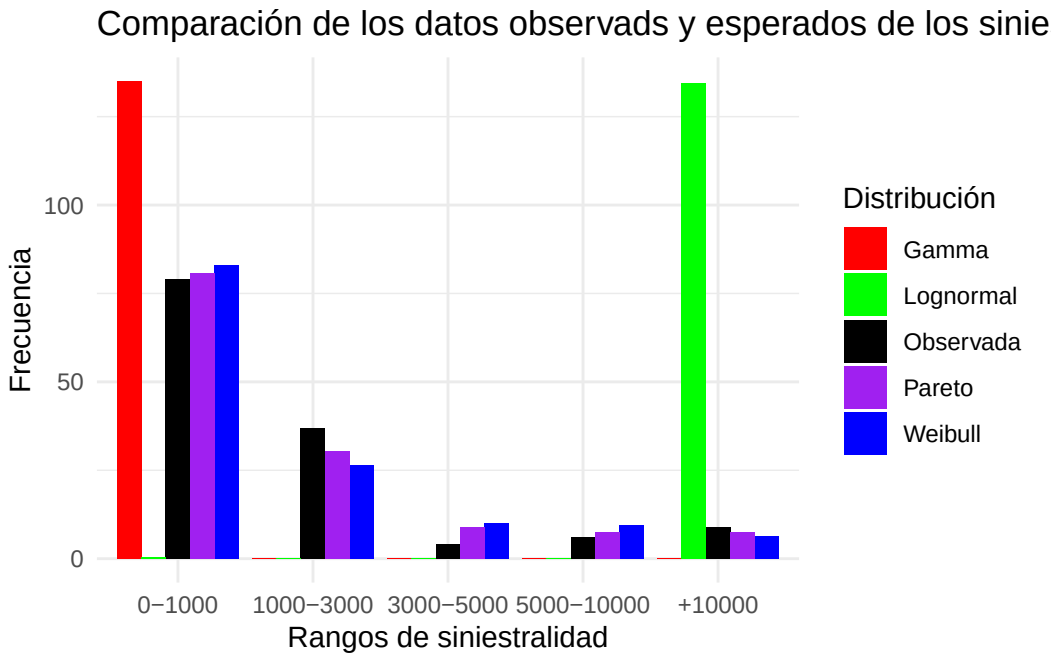
```r
(chisq_test_lognormal <- chisq.test(x = observed, p = expected_lognormal / sum(expected_lo
```

```
    Chi-squared test for given probabilities

data:  observed
X-squared = 29969, df = 4, p-value < 2.2e-16
```

```r
(chisq_test_pareto <- chisq.test(x = observed, p = expected_pareto / sum(expected_pareto),
```

```
    Chi-squared test for given probabilities

data:  observed
X-squared = 4.5808, df = 4, p-value = 0.3331
```

```r
# Crear un data frame con las frecuencias observadas y esperadas
comparacion <- data.frame(
  Rango = rep(sevDatos$Rango, 5),
  Frecuencia = c(sevDatos$Frecuencia, expected_weibull, expected_gamma, expected_lognormal
  Tipo = rep(c("Observada", "Weibull", "Gamma", "Lognormal", "Pareto"), each = nrow(sevDat
)

# Graficar las frecuencias observadas y esperadas
ggplot(comparacion, aes(x = Rango, y = Frecuencia, fill = Tipo)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Comparación de los datos observads y esperados de los siniestros por model
       x = "Rangos de siniestralidad",
       y = "Frecuencia",
       fill = "Distribución") +
  theme_minimal() +
  scale_fill_manual(values = c("Observada" = "black", "Weibull" = "blue", "Gamma" = "red",
```



Con lo cual evaluando las pruebas de bondad de ajuste, la gráfica comparativa y la métrica de AIC, notamos que el modelo que mejor se ajusta a los datos es el modelo de Pareto, por lo que es el modelo que usaremos para representar siniestralidad.

Prima de riesgo

Supuestos

El modelo de pérdida agregada para calcular la prima de riesgo promedio y la cuota de riesgo fue el siguiente:

$$S = \sum_{i=1}^{N} V_i W_i$$

Donde

- V_i es el valor de los instrumentos que estaríamos asegurando en nuestra cartera y suponemos que por evento es aleatorio con una distribución uniforme según los rangos que aseguraremos en la cartera (0-100000).
- W_i es la severidad (porcentaje de la siniestralidad con respecto al valor del intrumentos) de los siniestros ocurridos. Esta variable es aleatoria.

- N es el número de eventos que ocurrirán en el año. E igualmente es una variable aleatoria.

Suponemos que las 3 variables son independientes entre sí.

Dado este modelo de riesgo colectivo presentado, procedemos a calcular la prima de riesgo promedio y la cuota de riesgo.

La prima de riesgo promedio para este seguro de daños de instrumentos musicales se calcula como la esperanza del modelo:

$$E[S] = E[\sum_{i=1}^{N} V_i W_i] = E[N]E[V]E[W] = E[N]E[VW]$$

Donde $E[N]$ es la esperanza del número de eventos, y $E[VW] = E[V]E[W]$ es la siniestralidad esperada que tendríamos para una cartera equilibrada de instrumentos musicales.

Esta prima de riesgo promedio es lo que tendríamos que cobrar a todos los usuarios asegurados en nuestra cartera; sin embargo, para ser justos con los asegurados, tendríamos que cobrar una prima de acuerdo con el valor de los instrumentos que se cubren en la cartera, por lo que conviene calcular una cuota de riesgo que multiplicado por el valor del instrumento asegurado, nos de la prima que debemos cobrar.

Con lo cual, si suscribimos equilibradamente nuestros riesgos, tendriamos que cobrar una cuota de riesgo a cada instrumento asegurado de $E[N]E[W] = E[N]E[VW]/E[V]$. Donde $E[V]$ es el valor esperado de los instrumentos asegurados en nuestra cartera equilibrada. De esta forma, cobraríamos una prima justa, y además seguiríamos cubriendo los mismos supuestos del modelo de riesgo colectivo. Cabe recalcar que este cambio solo es posible si se suscriben riesgos que en promedio sea $E[V]$.

Ahora bien, para calcular esta prima de riesgo recurrimos numéricamente a simular los valores de nuestro modelo de riesgo colectivo, de acuerdo con modelos ajustados de la sección anterior para las variables aleatorias N y VW.

En este caso hay que notar, que para evitar muchas siniestralidades pequeñas se ha decidido que el deducible sea de 1000, coaseguro del 10%, y que el límite de la aseguradora sea de 100000.

Con lo cual, para nosotros nuestro riesgo de siniestralidad está medido de acuerdo con la siguiente variable $VW = \min((X - 1000)_+, 100000)$

Cálculos

Para poder calcular la esperanza utilizaremos técnicas avanzadas de simulación. En este caso utilizaremos el método de bootstrap paramétrico para calcular la esperanza de la siniestralidad.

```r
library(fitdistrplus)
library(actuar)  # Para la distribución Pareto
library(parallel)
set.seed(191654)
# Supuestos de los parámetros calculados anteriormente
deducible <- 1000    # Deducible para ajustar los valores de VW
lambda_poisson <- lambda_poisson  # Tasa de la distribución Poisson
alpha_pareto <- params_pareto$par[1]    # Parámetro de forma de la distribución Pareto
beta_pareto <- params_pareto$par[2]     # Parámetro de escala de la distribución Pareto

limAseg <- 80000  # Límite de la aseguradora
# Número de simulaciones
M <- 10000
B <- 10000

# Función de simulación optimizada
simular_S <- function(lambda_poisson, alpha_pareto, beta_pareto, deducible, M) {
  # 1. Simulaciones de N
  N <- rpois(M, lambda_poisson)

  # 2. Simulaciones de VW
  VW <- lapply(N, function(n) {
    if (n == 0) return(0)
    valores <- rpareto(n, shape = alpha_pareto, scale = beta_pareto)
    valores[valores > deducible] <- valores[valores > deducible] - deducible
    valores[valores <= deducible] <- 0
    valores[valores > limAseg] <- limAseg
    return(valores*.95)
  })

  # 3. Construcción de M simulaciones de S
  S <- sapply(VW, sum)

  VW <- unlist(VW)

  return(list(S = S, N = N, VW = VW))
}

# Función para ejecutar el proceso en paralelo
simular_bootstrap <- function(iter, lambda_poisson, alpha_pareto, beta_pareto, deducible,
  sim_data <- simular_S(lambda_poisson, alpha_pareto, beta_pareto, deducible, M)
```

```
  mean(sim_data$S)
}

# Configurar clúster para paralelización
cl <- makeCluster(detectCores() - 1)
clusterExport(cl, c("lambda_poisson", "alpha_pareto", "beta_pareto", "deducible", "M", "si

# Ejecutar simulaciones en paralelo
promedios_S <- parSapply(cl, 1:B, simular_bootstrap, lambda_poisson, alpha_pareto, beta_pa

# Detener clúster
#stopCluster(cl)

# Resultados
mean_promedio_S <- mean(promedios_S)
error_estandar_S <- sd(promedios_S)

# Mostrar resultados
cat("Promedio de S (Plugin de la esperanza de S):", mean_promedio_S, "\n")
```

```
Promedio de S (Plugin de la esperanza de S): 1854.658
```

```
cat("Error estándar bootstrap del promedio de S:", error_estandar_S, "\n")
```

```
Error estándar bootstrap del promedio de S: 79.89301
```

Es decir, la cuota de riesgo que tendríamos que cobrar a cada instrumento asegurado en nuestra cartera equilibrada sería de

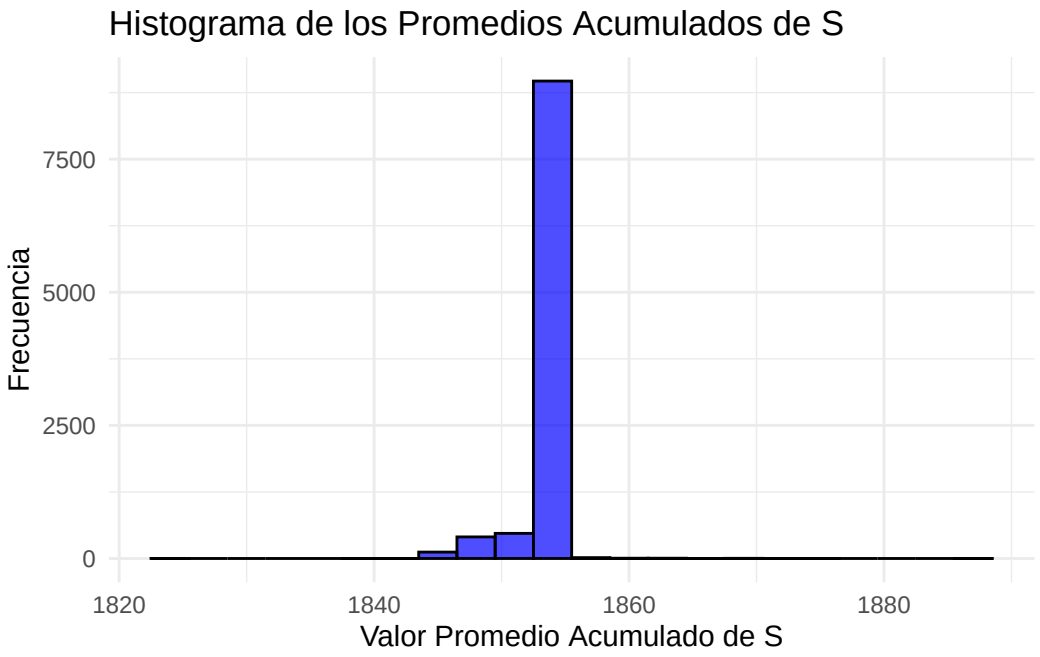
$$E[VW]/E[V] = \frac{1854.6584288}{2.4207143 \times 10^4}$$

0.0766162.

Para terminar de validar el ajuste presentamos el histograma de la distribución boostrap de la siniestralidad esperada agregada.

```
# Calcular los valores acumulados de los promedios de S
promedios_S_acumulados <- cumsum(promedios_S) / seq_along(promedios_S)
```

```r
library(ggplot2)

# Crear un data frame con los valores acumulados de los promedios de S
df_acumulados <- data.frame(
  Iteracion = seq_along(promedios_S_acumulados),
  Promedio_Acumulado_S = promedios_S_acumulados
)

# Crear el histograma para los valores acumulados
(plot_acumulado <- ggplot(df_acumulados, aes(x = Promedio_Acumulado_S)) +
  geom_histogram(binwidth = 3, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histograma de los Promedios Acumulados de S",
       x = "Valor Promedio Acumulado de S",
       y = "Frecuencia") +
  theme_minimal())
```




```r
# Guardar la gráfica en un archivo JPG
ggsave(filename = "histograma_promedios_acumulados_S.jpg", plot = plot_acumulado, device =
```

RCS

Ahora dado los resultados anteriores debemos de calcular el requerimiento de capital de solvencia (RCS) para la aseguradora. Para ello, debemos de calcular 3 componentes:

$$\mathbf{RCS} = \mathbf{RCS}_T + \mathbf{RCS}_F + \mathbf{RCS}_{Op}$$

Donde:

- $\mathbf{RCS}_T$ es el requerimiento de capital de solvencia técnico, que se calcula como el exceso del percentil 99.5 de la distribución de la siniestralidad esperada agregada con respecto a lo esperado. Es decir, $\mathbf{RCS}_T = q_{0.995}(S) - E(S)$. Esta fórmula es muy simple porque solo tenemos un producto y es lo máximo que podríamos perder en un año con una confianza del 99.5%.
- $\mathbf{RCS}_F$ es el requerimiento de capital de solvencia financiero. En este caso es cero, porque suponemos que no tendremos riesgos financieros a un año.
- $\mathbf{RCS}_{Op}$ es el requerimiento de capital de solvencia operativo, que se calcula como el 15% de la prima de riesgo promedio.

Por lo tanto, para el cálculo del RCS, procedemos a calcular el RCS técnico y el RCS operativo.

```r
# Calcular el RCS técnico
# Número de simulaciones
M <- 10000
B <- 10000

# Función de simulación optimizada
simular_S <- function(lambda_poisson, alpha_pareto, beta_pareto, deducible, M) {
  # 1. Simulaciones de N
  N <- rpois(M, lambda_poisson)

  # 2. Simulaciones de VW
  VW <- lapply(N, function(n) {
    if (n == 0) return(0)
    valores <- rpareto(n, shape = alpha_pareto, scale = beta_pareto)
    valores[valores > deducible] <- valores[valores > deducible] - deducible
    valores[valores <= deducible] <- 0
    valores[valores > limAseg] <- limAseg
    return(valores*.95)
  })
```

```r
  # 3. Construcción de M simulaciones de S
 # Calcular el percentil 99.5 de S
  S <- sapply(VW, sum)


  VW <- unlist(VW)

  return(list(S = S, N = N, VW = VW))
}

# Función para ejecutar el proceso en paralelo
simular_bootstrap <- function(iter, lambda_poisson, alpha_pareto, beta_pareto, deducible,
  sim_data <- simular_S(lambda_poisson, alpha_pareto, beta_pareto, deducible, M)
  quantile(sim_data$S, 0.995)
}

# Configurar clúster para paralelización
cl <- makeCluster(detectCores() - 1)
clusterExport(cl, c("lambda_poisson", "alpha_pareto", "beta_pareto", "deducible", "M", "si

# Ejecutar simulaciones en paralelo
cuantiles_S <- parSapply(cl, 1:B, simular_bootstrap, lambda_poisson, alpha_pareto, beta_pa

# Detener clúster
#stopCluster(cl)

# Resultados
mean_cuantil_S <- mean(cuantiles_S)
error_estandar_S <- sd(cuantiles_S)

# Mostrar resultados
cat("Promedio de S (Plugin de la esperanza de S):", mean_cuantil_S, "\n")
```

```
Promedio de S (Plugin de la esperanza de S): 74522.09
```

```r
cat("Error estándar bootstrap del promedio de S:", error_estandar_S, "\n")
```

```
Error estándar bootstrap del promedio de S: 3264.843
```

```r
# Intervalos:

# Calcular los intervalos de confianza bootstrap utilizando cuantiles
cuantiles_S_Int <- quantile(cuantiles_S, c(0.025, 0.5, .75,0.975))

# Extraer los cuantiles
(cuantil_2.5 <- cuantiles_S_Int[1])
```

```
   2.5%
64136.3
```

```r
(cuantil_50 <- cuantiles_S_Int[2])
```

```
  50%
76000
```

```r
(cuantil_75 <- cuantiles_S_Int[3])
```

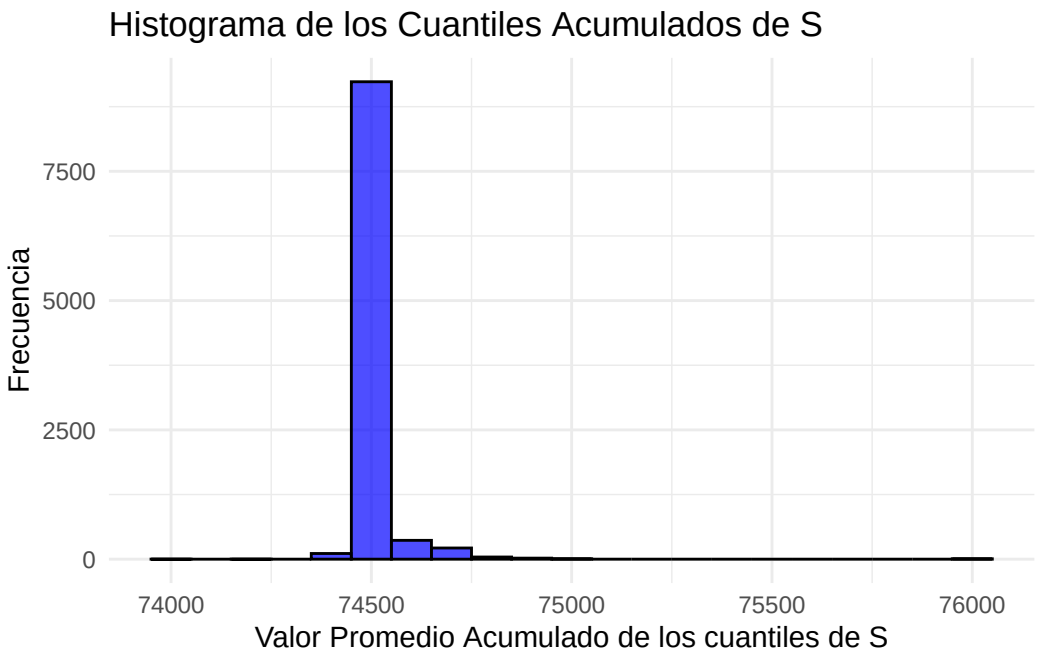
```
  75%
76000
```

```r
(cuantil_97.5 <- cuantiles_S_Int[4])
```

```
97.5%
76000
```

```r
#  Gráfica
cuantiles_S_acumulados <- cumsum(cuantiles_S) / seq_along(cuantiles_S)

# Crear un data frame con los valores acumulados de los promedios de S
df_acumulados <- data.frame(
  Iteracion = seq_along(cuantiles_S_acumulados),
  Promedio_Acumulado_S = cuantiles_S_acumulados
)

# Crear el histograma para los valores acumulados
(plot_acumulado <- ggplot(df_acumulados, aes(x = Promedio_Acumulado_S)) +
```

```
  geom_histogram(binwidth = 100, fill = "blue", color = "black", alpha = 0.7) +
  labs(title = "Histograma de los Cuantiles Acumulados de S",
       x = "Valor Promedio Acumulado de los cuantiles de S",
       y = "Frecuencia") +
  theme_minimal())
```




```
# Guardar la gráfica en un archivo JPG
ggsave(filename = "histograma_cuantiles_acumulados_S.jpg", plot = plot_acumulado, device =
```

Es decir, el RCS técnico que tendríamos que tener para asegurar los riesgos de siniestralidad esperada agregada es de 7.2667429×10^4 en promedio por cada póliza.

Por otro lado, el RCS operativo que tendríamos que tener para asegurar los riesgos de operación es de 278.1987643 por cada póliza.

Por lo tanto, el RCS total que tendríamos que tener para asegurar los riesgos de siniestralidad y operativos es de 7.2945628×10^4 por cada póliza.

Tarificación

Para la tarificación de los seguros de instrumentos musicales, se propone una tarifa de acuerdo con la cuota de riesgo calculada anteriormente, pero agregamos gastos administrativos del

10% de la cuota de riesgo, utilidad del 20%, y un margen de seguridad del 20% de la cuota de riesgo.

Por lo tanto, la cuota de tarifa que proponemos para el seguro de instrumentos musicales es de 0.1532323 por cada instrumento asegurado.

Al final, nos daría una tarifa promedio de 3709.3168576 por cada instrumento asegurado.

RRC

Para la RRC, debemos reservar por un año un porcentaje por cada póliza vendida, este porcentaje se calcula de la siguiente forma para cada póliza vendida:

$$RRC\% = \frac{\text{primaRiesgo}}{\text{primaTarifa}} = 0.5$$

Suponemos que en todos los años las pólizas se adquieren uniformemente a lo largo del año, y se devengan uniformemente por el plazo del seguro renovable, que es un año. Por lo que en un año, de todo lo que se venda en pólizas se debe reservar la mitad de la prima de riesgo promedio.

ROPC

Como no tenemos experiencia en el mercado, proponemos un ROPC del 100% de la prima de riesgo promedio, es decir, 0.5 por cada póliza vendida. Ya cuando generemos experiencia, podremos ajustar este parámetro.