Clase Modelado 1

2024-05-20

Modelos

```
library(dplyr)
```

```
Attaching package: 'dplyr'

The following objects are masked from 'package:stats':

    filter, lag

The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(purrr)
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```r
library(fitdistrplus)
```

```
Loading required package: survival
```

```r
library(actuar)
```

```
Attaching package: 'actuar'

The following objects are masked from 'package:stats':

    sd, var

The following object is masked from 'package:grDevices':

    cm
```

```r
# Datos
# Los que no son 10,000-50,000
# Crear un 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, 7)
)

frecDatos <- data.frame(
  Rango = c("0--1", "2--3", "3--4"),
  Completa = c(56, 12, 2)
)
```

Modelos de frecuencia

```r
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
```

```r
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
```

```r
# 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]]
})



# 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
```

Modelos de severidad

```
# Cargar el paquete readxl
library(readxl)
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)

# Modificamos la lista
mydata <- mydata |>
  filter(Intermedio>500)
mean(mydata$Intermedio)
```

```
[1] 24904.41
```

Modelos de siniestralidad

```
library(fitdistrplus)
library(ggplot2)
# deducible <- 1000
# sevDatos <- sevDatos |>
#   filter(Rango != "0-1000")

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

```r
# 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$Frecuencia[i] * log(plnorm(log(limites[[i]][2]), meanlog, sdlog) - plnorm(log
  }))#+ sum(sevDatos$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
```

```r
ajustar_distribucion <- function(logverosimilitud, start_params) {
  fit <- optim(start_params, logverosimilitud)
  return(fit)
}

# Ajustar las distribuciones
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] 296.8985
```

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

```
[1] 300.6238
```

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

```
[1] 291.4332
```

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

```
[1] 292.1386
```

```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] 133
```

```
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_weibull
chisq_test_gamma <- chisq.test(x = observed, p = expected_gamma / sum(expected_gamma), res
chisq_test_lognormal <- chisq.test(x = observed, p = expected_lognormal / sum(expected_log
chisq_test_pareto <- chisq.test(x = observed, p = expected_pareto / sum(expected_pareto),

# Mostrar los resultados de las pruebas chi cuadrado
print(chisq_test_weibull)
```

```
  Chi-squared test for given probabilities
```

```
data:  observed
X-squared = 8.6885, df = 4, p-value = 0.06938
```

```
print(chisq_test_gamma)
```

```
    Chi-squared test for given probabilities

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

```
print(chisq_test_lognormal)
```

```
    Chi-squared test for given probabilities

data:  observed
X-squared = 3.4772, df = 4, p-value = 0.4814
```

```
print(chisq_test_pareto)
```

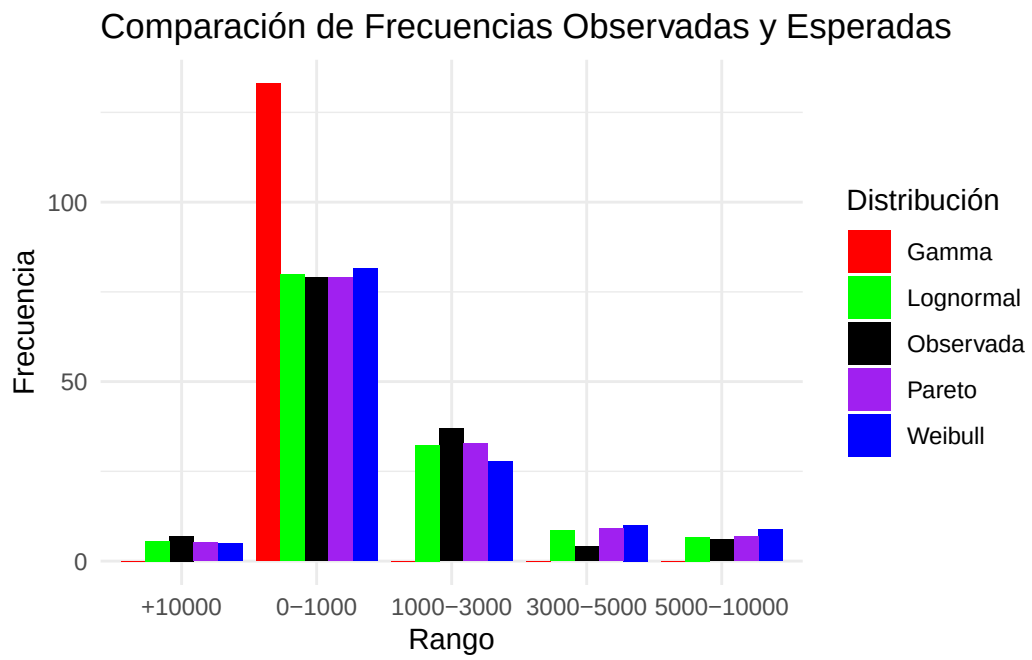
```
    Chi-squared test for given probabilities

data:  observed
X-squared = 4.0556, df = 4, p-value = 0.3985
```

```
# 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 Frecuencias Observadas y Esperadas",
       x = "Rango",
       y = "Frecuencia",
       fill = "Distribución") +
  theme_minimal() +
  scale_fill_manual(values = c("Observada" = "black", "Weibull" = "blue", "Gamma" = "red",
```



Pareto

```
  (esperanza_pareto <-  params_pareto$par[2] / (params_pareto$par[1] - 1))
```

```
[1] 2735.197
```

```
  (severidad <- esperanza_pareto/mean(unique(mydata$Intermedio)))
```

```
[1] 0.09472543
```

Prima de riesgo promedio

```
(primaRiesgoProm <- esperanza_pareto*lambda_poisson)
```

```
[1] 2371.741
```

Prima de riesgo

```
(primaRiesgo <- severidad*lambda_poisson)
```

```
[1] 0.08213821
```

RCS