Bondad de ajuste.

Para evaluar qué tan bien representan nuestros datos a la distribuciones paramétricas propuestas, utilizamos distintas herramientas de bondad de ajuste.

Herramientas gráficas

- QQ-plot.
- Histograma/Densidad empírica.
- Función de distribución empírica.

Gráfico cuantil-cuantil (QQ-plot). La idea es graficar los pares de coordenadas del cuantil teórico (del modelo propuesto y ya estimado) contra el cuantil empírico:

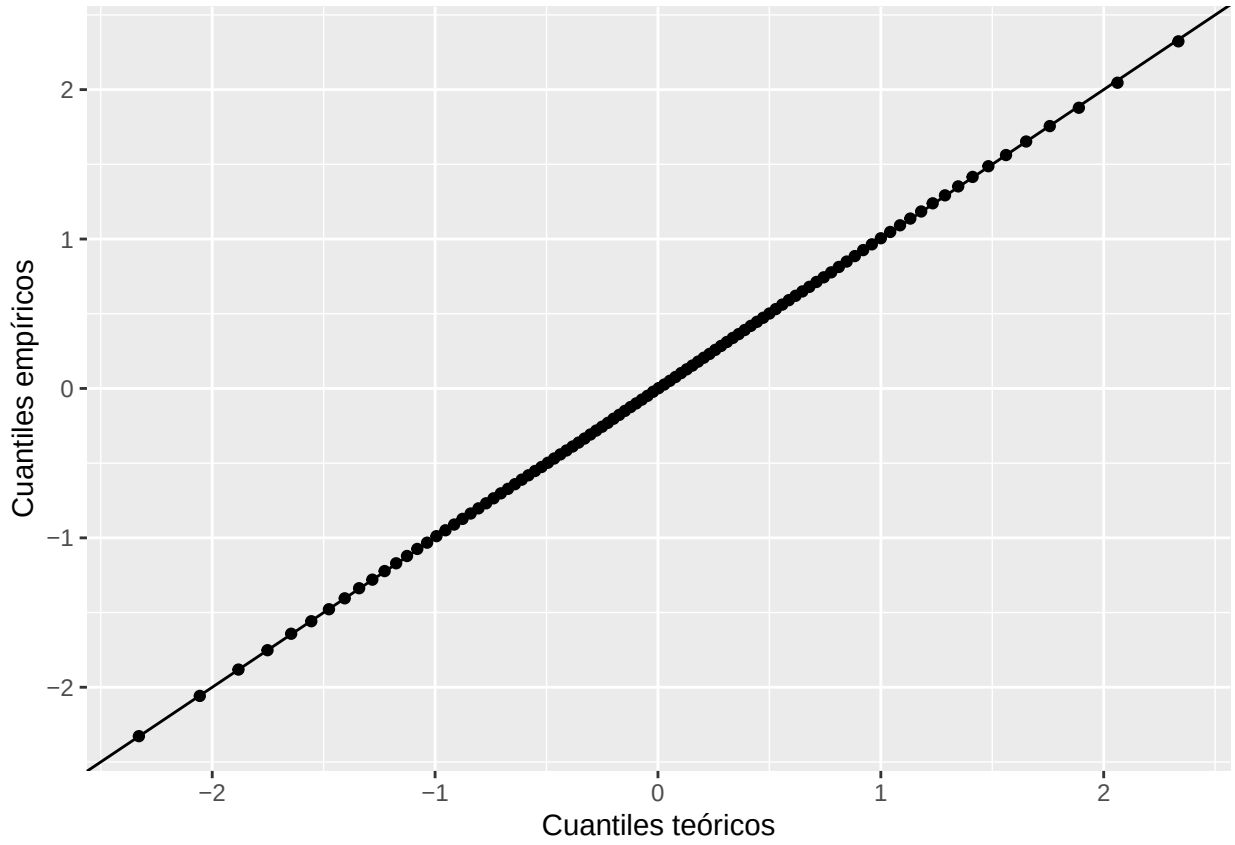
$$< F_X^{-1}(\frac{1}{n}), EF_X^{-1}(\frac{1}{n}) >, < F_X^{-1}(\frac{2}{n}), EF_X^{-1}(\frac{2}{n}) >, ..., < F_X^{-1}(\frac{n-1}{n}), EF_X^{-1}(\frac{n-1}{n}) >$$

Esto no es más que los pares de cuantiles desde $\frac{1}{n}$ hasta $\frac{n-1}{n}$

La n estará en función del número de cuantiles que busquemos calcular.

Ejemplo:

```
library(tidyverse)
muestra <- rnorm(1e5)

cuantiles <- 1:99/100

# Cuantiles empíricos:
q_emp <- quantile(muestra, cuantiles)

# Cuantiles teóricos: proponiendo que los datos vienen de una distribución normal.
q_teo <- qnorm(p = cuantiles, mean = mean(muestra), sd = sd(muestra))

library(ggplot2)

ggplot()+
  geom_point(aes(q_teo, q_emp))+
  geom_abline(intercept = 0, slope = 1)+
  labs(x = "Cuantiles teóricos",
       y = "Cuantiles empíricos")
```



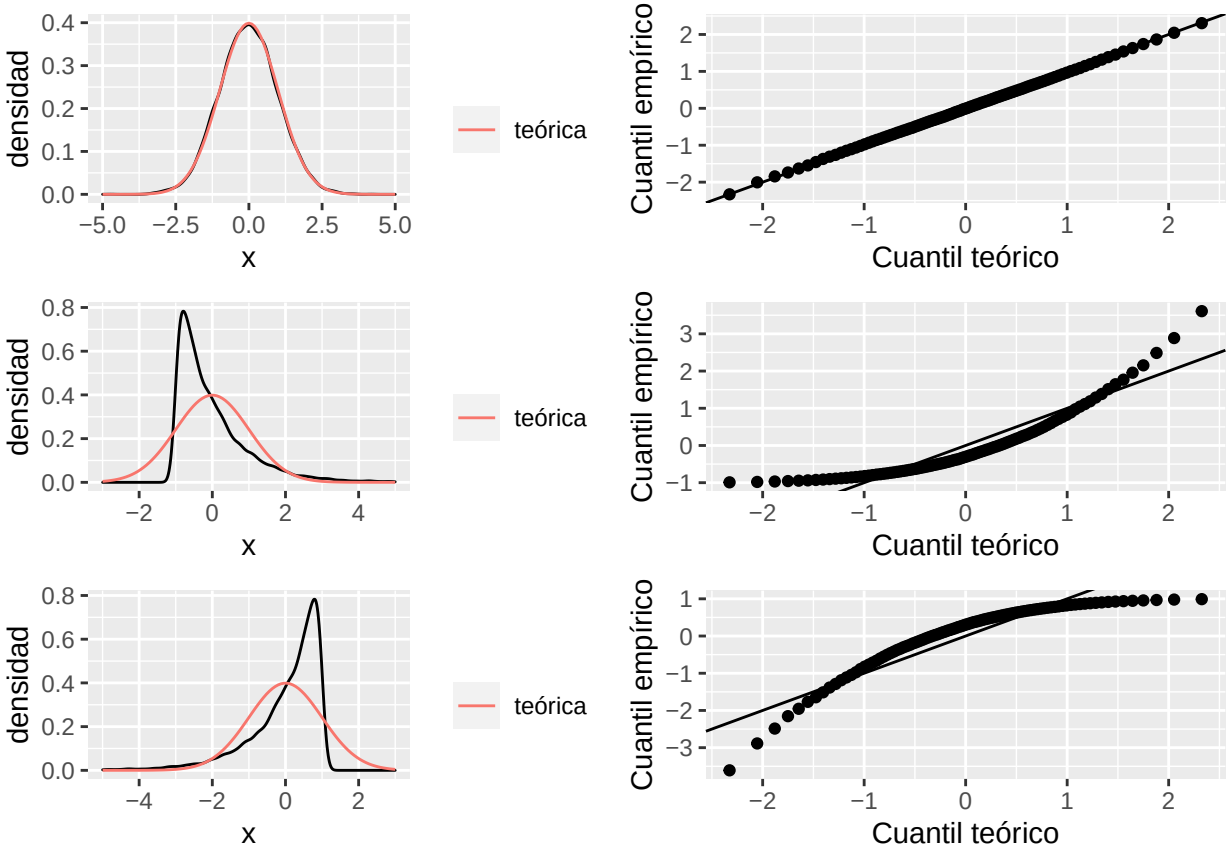
En este caso, prácticamente todos los puntos caen sobre la línea en la que $x = y$, puesto que sabemos que la muestra viene de una distribución normal. Generalmente no observaremos esto, pero si fuera el caso, quiere decir que $F_X(x) = EF_X(x)$.

Normalmente, nos encnontraremos con algún caso como los siguientes:

Ilustración qq-plots:

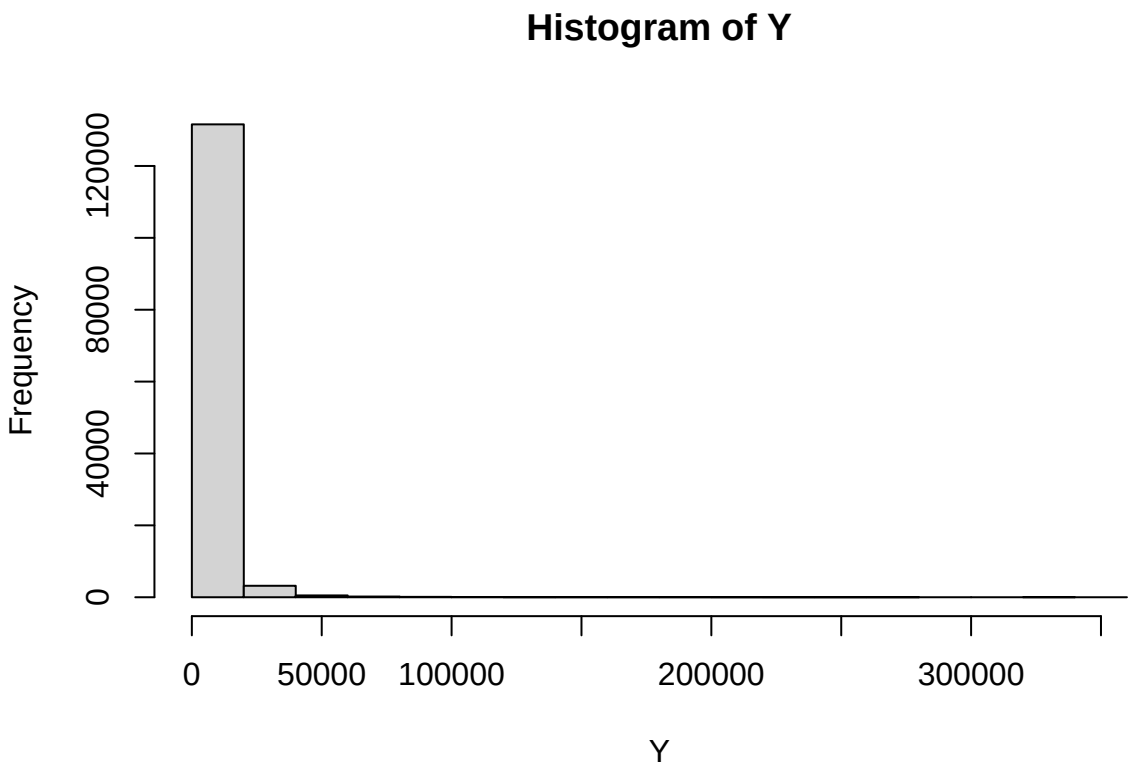
```
library(ggplot2)
library(gridExtra)

x <- rexp(1e4)

ps <- 1:99/100

cd1 <- ggplot()+
  geom_density(aes(scale(x)))+
  geom_function(fun = function(x){dnorm(x)}, aes(color = "teórica"))+
  xlim(-3,5)+
  labs(x = "x", y = "densidad", color = "")

ci1 <- ggplot()+
  geom_density(aes(scale(-x)))+
  geom_function(fun = function(x){dnorm(x)}, aes(color = "teórica"))+
  xlim(-5,3)+
```

```
  labs(x = "x", y = "densidad", color = "")

cs1 <- ggplot()+
  geom_density(aes(rnorm(1e4)))+
  geom_function(fun = function(x){dnorm(x)}, aes(color = "teórica"))+
  xlim(-5,5)+
  labs(x = "x", y = "densidad", color = "")

cd2 <- ggplot()+
  geom_point(aes(qnorm(ps),quantile(scale(x), ps)))+
  geom_abline(slope = 1, intercept = 0)+
  labs(x = "Cuantil teórico",
       y = "Cuantil empírico")


ci2 <- ggplot()+
  geom_point(aes(qnorm(ps),quantile(scale(-x), ps)))+
  geom_abline(slope = 1, intercept = 0)+
  labs(x = "Cuantil teórico",
       y = "Cuantil empírico")

cs2 <- ggplot()+
  geom_point(aes(qnorm(ps),quantile(rnorm(1e4), ps)))+
  geom_abline(slope = 1, intercept = 0)+
  labs(x = "Cuantil teórico",
       y = "Cuantil empírico")

grid.arrange(cs1, cs2, cd1, cd2, ci1, ci2, ncol = 2)
```



Ejemplo con datos de gasto en transporte:

```
bd <- read.csv("D:/UAG/Modelación actuarial/2022/clases/14. Bondad de ajuste/muestratransporte.csv")

Y <- bd$x
hist(Y)
```




```
summary(Y)
```

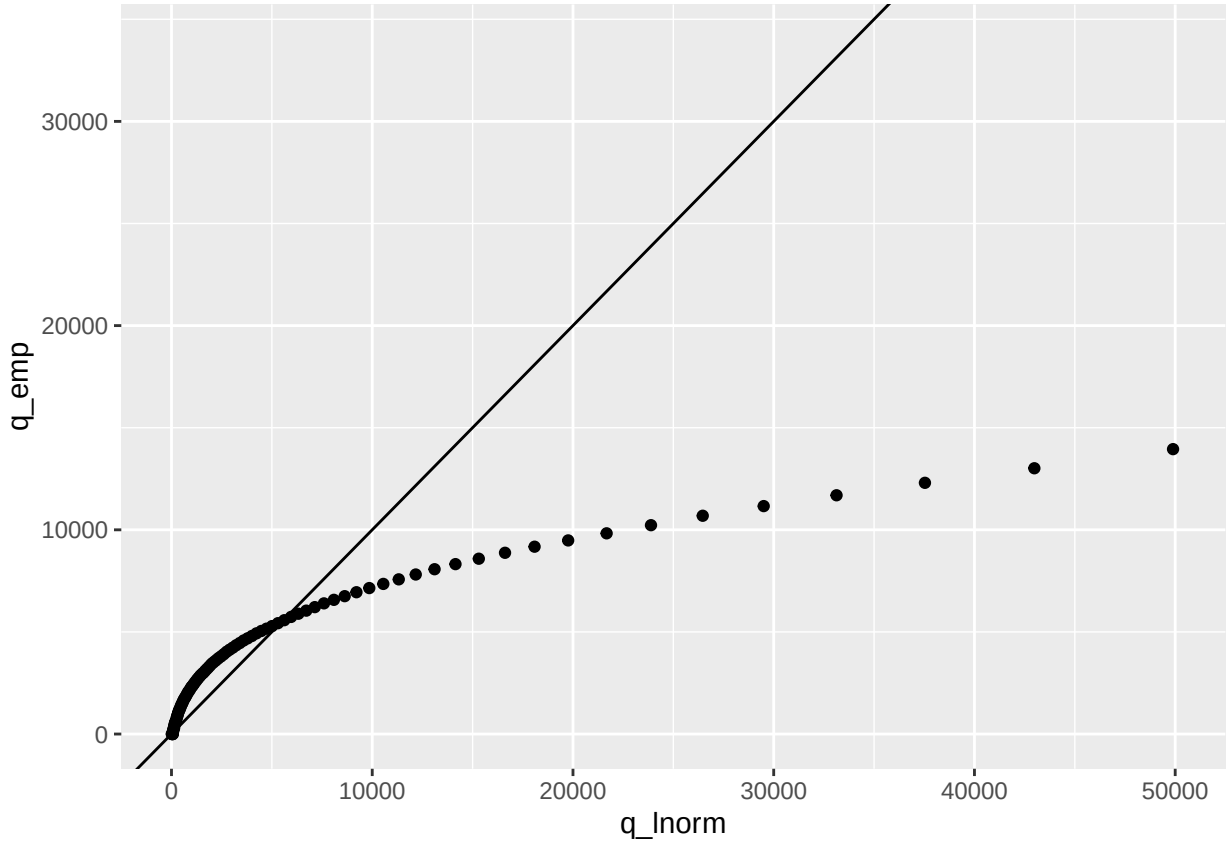
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0    1500    3613    5650    6943  348729
```

```
# Propondremos las siguientes distribuciones paramétricas para modelar la variable:

# Lognormal y Gamma. Con estas distribuciones, no podemos usar valores igual que cero, por lo que model

W <- Y+1

# Ajustamos una lognormal:
mu_MLE <- mean(log(W))
sigma_MLE <- sd(log(W))

# Ajustamos una Gamma:
## Primero estimamos por el método de momentos.
k_MM <- (mean(W)^2)/var(W)
theta_MM <- var(W)/mean(W)
## Ahora por MLE con los valores iniciales definidos por el MM.
gamma_MLE <- optim(par = c(k_MM, theta_MM),
                   muestra = W,
                   fn = function(par, muestra){
                     -sum(
                       log(
```

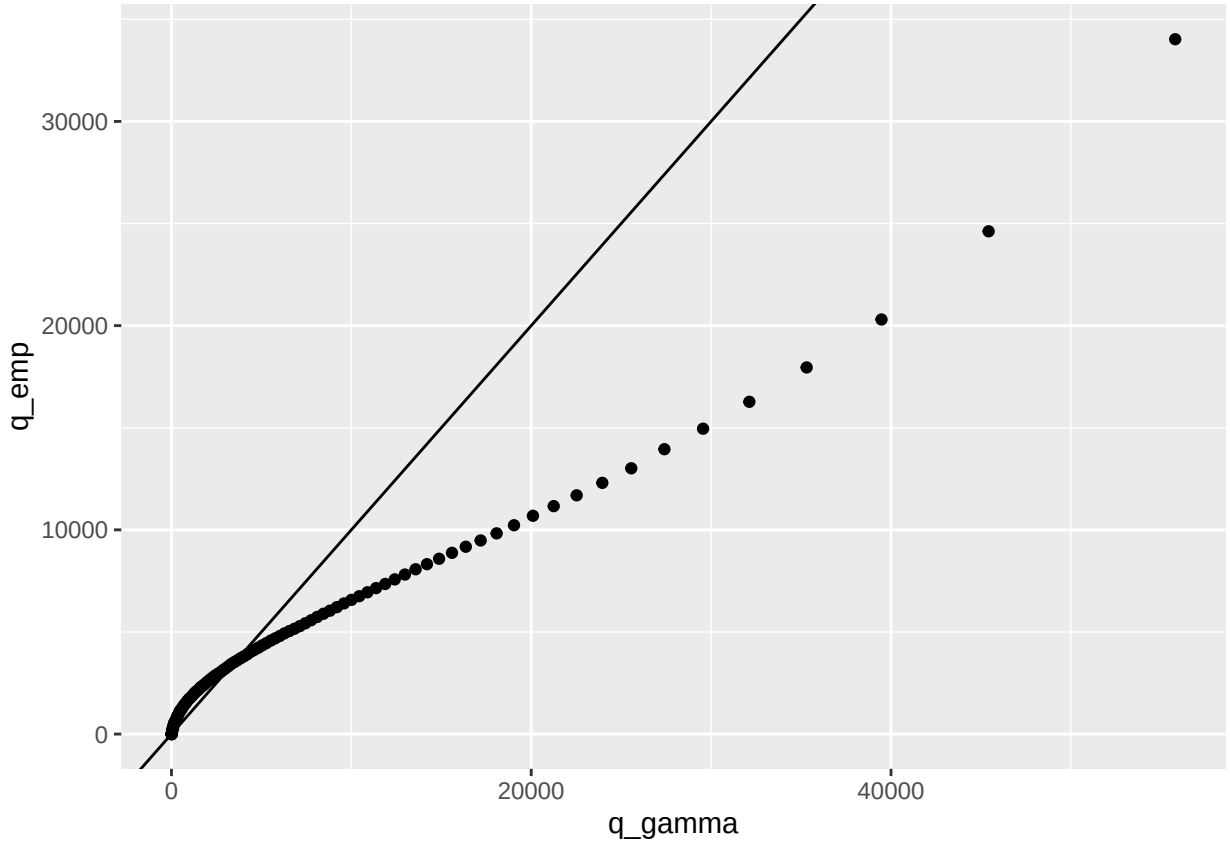
```
                        dgamma(muestra,
                               shape = par[1],
                               scale = par[2])
                      )
                    )
                  }, method = "L-BFGS-B",
                  lower = c(1e-4, 1e-4),
                  upper = c(Inf, Inf))

k_MLE <- gamma_MLE$par[1]
theta_MLE <- gamma_MLE$par[2]
```

Ahora, podemos evaluar gráficamente el ajuste de las distribuciones:

```
cuantiles <- 1:99/100
q_emp <- quantile(W, cuantiles)
q_lnorm <- qlnorm(cuantiles, mu_MLE, sigma_MLE)
q_gamma <- qgamma(cuantiles, shape = k_MLE, scale = theta_MLE)

# QQ-plot para la lognormal:
ggplot()+
  geom_point(aes(q_lnorm, q_emp))+
  geom_abline(intercept = 0, slope = 1)+
  xlim(0, 50000)
```



```r
## QQ-plot para la gamma:
ggplot()+
  geom_point(aes(q_gamma, q_emp))+
  geom_abline(intercept = 0, slope = 1)
```

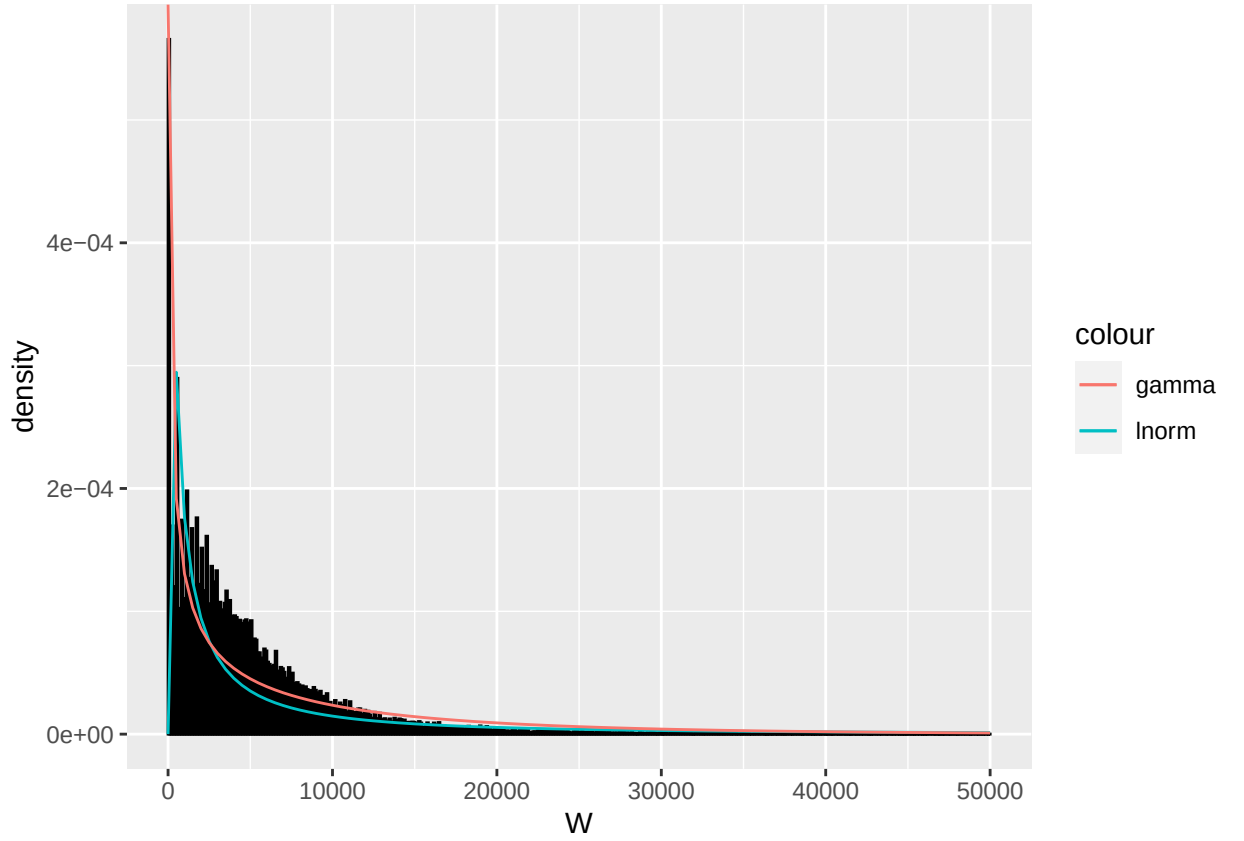


Vemos que, aunque ninguna de las dos distribuciones se ajustan muy bien, sobretodo en la cola derecha, la gamma se parece un poco más que la lognormal. Ambos modelos tienen cola derecha más pesada que la muestra.

Histograma/densidad empírica: Podemos ilustrar la útilidad de los histogramas de la siguiente manera:

```r
ggplot()+
  geom_histogram(aes(W, y = ..density..),
                 color = "black", boundary = 0,
                 bins = 500)+
  geom_function(aes(color = "lnorm"),
                fun = function(x){
                  dlnorm(x, mu_MLE, sigma_MLE)
                })+
  geom_function(aes(color = "gamma"),
                fun = function(x){
                  dgamma(x, shape = k_MLE,
                         scale = theta_MLE)
                })+
```

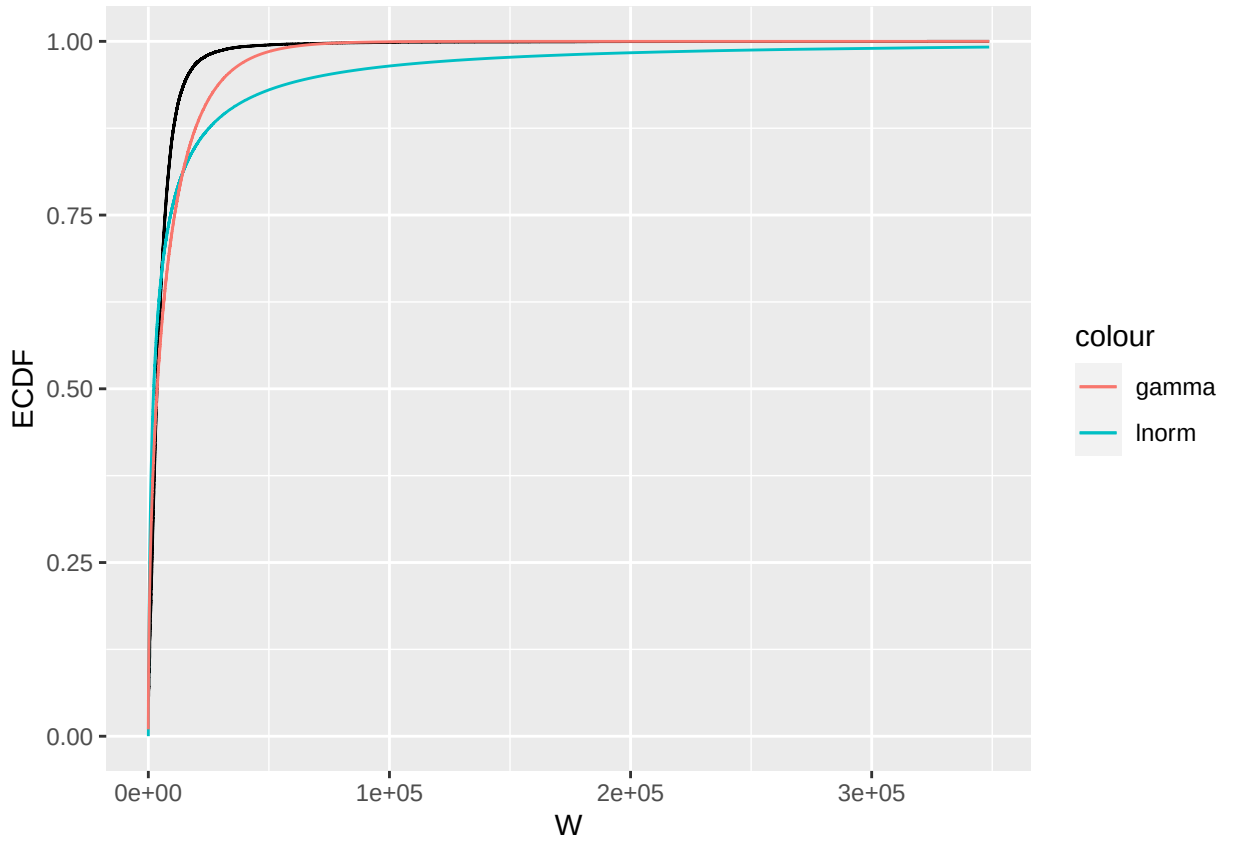
```
  xlim(0, 50e3)
```



Vemos que ambas funciones de densidad tienen un ajuste más o menos bueno hasta cierto punto, donde empiezan a diferir porque las distribuciones paramétricas propuestas tienen colas derechas más pesadas.

Función de distribución empírica vs. teórica

```
ECDF <- ecdf(W)(W)
CDF_lnorm <- plnorm(W, mu_MLE, sigma_MLE)
CDF_gamma <- pgamma(W, shape = k_MLE, scale = theta_MLE)

ggplot()+
  geom_step(aes(W, ECDF))+
  geom_line(aes(W, CDF_lnorm, color = "lnorm"))+
  geom_line(aes(W, CDF_gamma, color = "gamma"))
```



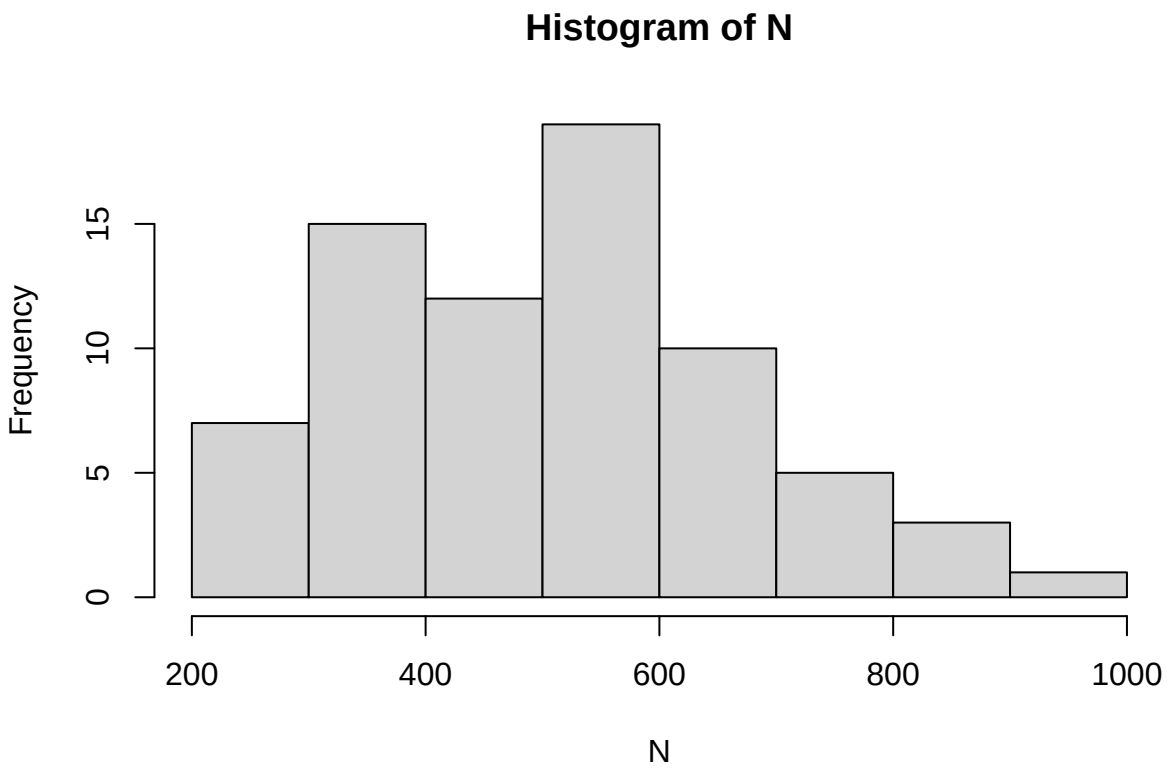
Ejemplo distribuciones discretas

Modelaremos el número de accidentes viales al mes, en el Área Metropolitana de Guadalajara entre 2015 y 2020.

```
# Primero obtenemos el vector de la variable que buscamos modelar:
bd_siniestros <- read.csv("siniestros_ZMG.csv", encoding = "UTF-8")

bd_siniestros <- bd_siniestros %>%
  group_by(anio, mes) %>%
  summarise(choques = n())

N <- bd_siniestros$choques

# Podemos explorar la variable:
summary(N)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   212.0   386.8   518.5   506.3   604.8   919.0
```

```
hist(N)
```




```r
var(N)/mean(N) # Si esta división fuera cercana a 1, probablemente la Poisson sería una distribución út
```

```
## [1] 51.08381
```

Para modelar la variable, usaremos la distribución de Poisson y la Binomial negativa.

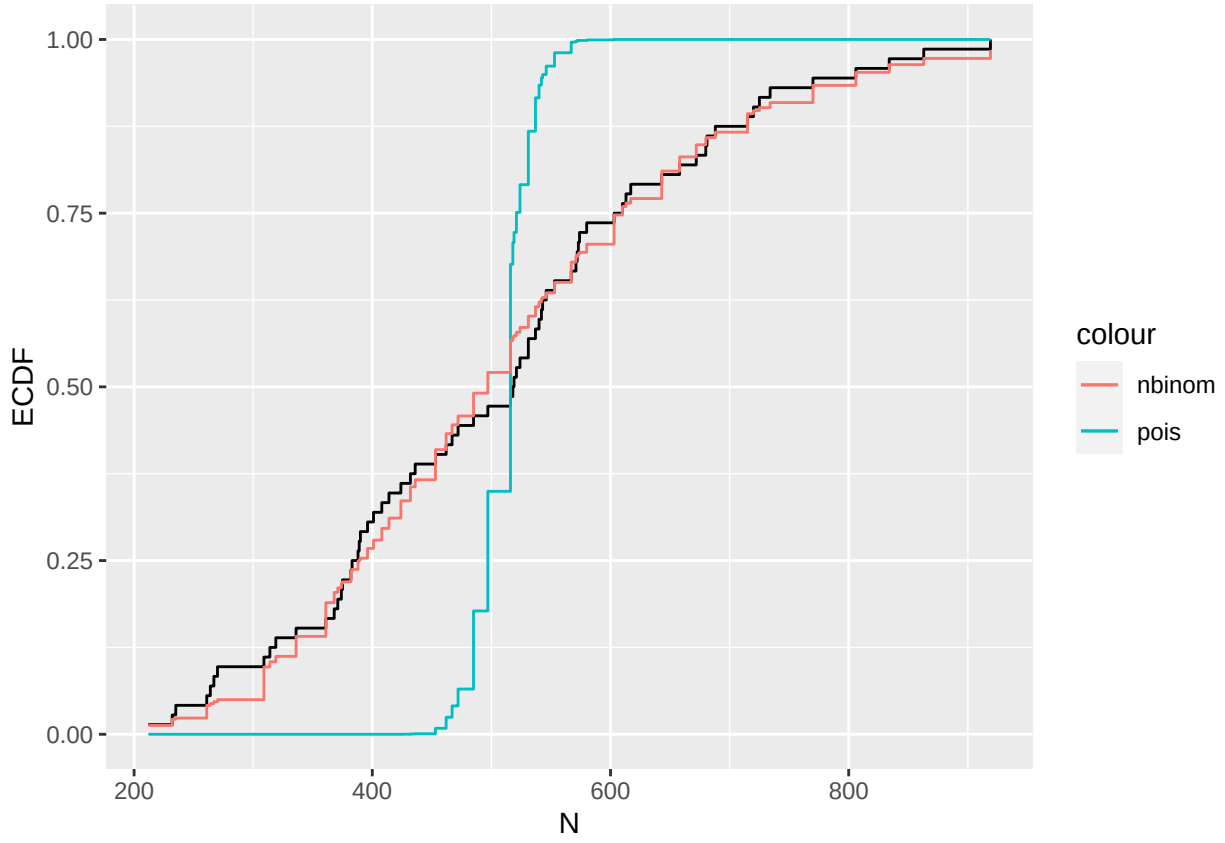
```r
# Poisson:
## Por máxima verosimilitud:
lambda_MLE = mean(N)

# Binomial negativa:

## Aproximamos los valores iniciales para la función optim con el método de momentos:
n <- length(N)
p_MM <- mean(N)/(var(N)*((n-1/n)))
r_MM <- mean(N)*p_MM/(1-p_MM)
## Usamos la función optim para obtener los estimadores por MLE:

nbinom_MLE <- optim(par = c(r_MM, p_MM),
                    datos = N,
                    fn = function(par, datos){
                      -sum(
                        log(
                          dnbinom(datos, par[1], par[2])
```

```
                        )
                      )
                    })
r_MLE <- nbinom_MLE$par[1]
p_MLE <- nbinom_MLE$par[2]
```

Ya con los modelos estimados, podemos compararlos con las herramientas gráficas vistas:

Función de distribución empírica vs. Teórica

```
CDF_pois <- ppois(N, lambda = lambda_MLE)
CDF_nbinom <- pnbinom(N, size = r_MLE, prob = p_MLE)
ECDF <- ecdf(N)(N)

ggplot()+
  geom_step(aes(N, ECDF))+
  geom_step(aes(N, CDF_pois, colour = "pois"))+
  geom_step(aes(N, CDF_nbinom, colour = "nbinom"))
```



Parece que, con este gráfico, la binomial negativa es mucha mejor opción que la Poisson.

Otra forma de visualizar y comparar nuestros datos con las distribuciones propuestas, para el caso de variables discretas, es con las proporciones observadas y las esperadas según los modelos:

```r
rangos <- cut(N, breaks = c(0, 300, 400, 500, 600, 700, Inf))
# Normalmente se busca que haya, por lo menos 5 valores por rango.
# Calculando las probabilidades observadas:
p_obs <- table(rangos)/n

# Calculando las probabilidades teóricas esperadas en los modelos:

p_pois <- c(ppois(300, lambda_MLE),
            ppois(400, lambda_MLE)-ppois(300, lambda_MLE),
            ppois(500, lambda_MLE)-ppois(400, lambda_MLE),
            ppois(600, lambda_MLE)-ppois(500, lambda_MLE),
            ppois(700, lambda_MLE)-ppois(600, lambda_MLE),
            1-ppois(700, lambda_MLE)
            )

p_nbinom <- c(pnbinom(300, r_MLE, p_MLE),
            pnbinom(400, r_MLE, p_MLE)-pnbinom(300, r_MLE, p_MLE),
            pnbinom(500, r_MLE, p_MLE)-pnbinom(400, r_MLE, p_MLE),
            pnbinom(600, r_MLE, p_MLE)-pnbinom(500, r_MLE, p_MLE),
            pnbinom(700, r_MLE, p_MLE)-pnbinom(600, r_MLE, p_MLE),
            1-pnbinom(700, r_MLE, p_MLE)
            )

cbind(p_obs, p_pois, p_nbinom)
```

```
##                p_obs       p_pois   p_nbinom
## (0,300]   0.09722222 2.159069e-23 0.08425585
## (300,400] 0.20833333 5.422602e-07 0.19278401
## (400,500] 0.16666667 4.003982e-01 0.25112834
## (500,600] 0.26388889 5.995779e-01 0.21407680
## (600,700] 0.13888889 2.336091e-05 0.13672765
## (700,Inf] 0.12500000 2.220446e-16 0.12102735
```

Así, vemos que los datos se parecen más a lo que veríamos con la binomial negativa. Es buena práctica no sólo comparar con un grupo de intervalos, sino con varios, para ver si siguen el mismo comportamiento, independientemente de la elección de estos valores.

Pruebas de bondad de ajuste

Existen distintas pruebas estadísticas que evalúan el ajuste de una muestra de datos sobre una distribución paramétrica, entre estas:

- Chi cuadrada de bondad de ajuste.
- Kolmogorov-Smirnov (KS).
- Anderson-Darling (AD).
- Jarque-Bera.

Prueba χ^2 de bondad de ajuste Esta prueba consiste en la comparación entre las probabilidades (o proporciones) empíricas con las teóricas, mediante un estadístico que sigue una distribución χ^2 bajo la hipótesis nula.

$H_0 : \hat{p} = p$

Para todas las probabilidades. También se puede pensar como que la función de distribución empírica es igual a la teórica.

$H_1 : \hat{p} \neq p$

En este tipo de pruebas, buscaremos no rechazar la hipótesis nula, para así poder afirmar con cierta confianza que nuestros datos vienen de la distribución paramétrica propuesta.

El estadístico de la prueba es el siguiente:

$Q_n = n \sum_{i=1}^{k} \frac{(\hat{p}_k - p_k)^2}{p_k}$

Donde $\hat{p}_k$ es la proporción empírica dentro del conjunto k y p_k es la proporción o probabilidad teórica de que existan valores en el conjunto k.

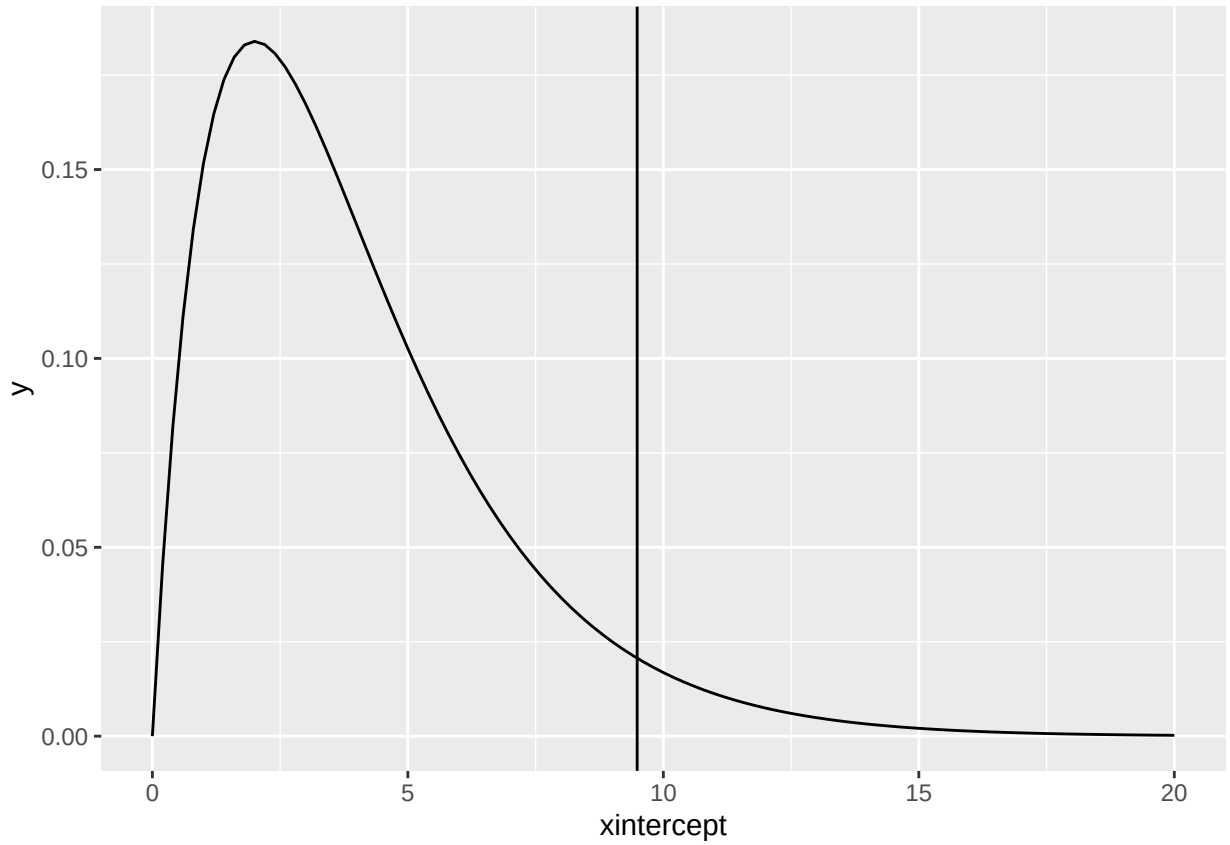
Si la hipótesis nula es verdadera, entonces el estadístico $Q_n \sim \chi^2_{(k-d-1)}$, donde k es el número de grupos y d es el número de parámetros estimados.

```
# Aprovechando que ya habíamos creado las tablas anteriormente:
n = length(N)
## Poisson:
Q_pois = n*sum(((p_obs-p_pois)^2)/p_pois)

## Obtenemos el cuantil 0.95 para una chi cuadrada con k-d-1 = 6 - 1 -1 = 4

qchisq(.95, df = length(p_obs)-1-1)
```

Ejemplo: Accidentes viales en Guadalajara:

```
## [1] 9.487729
```

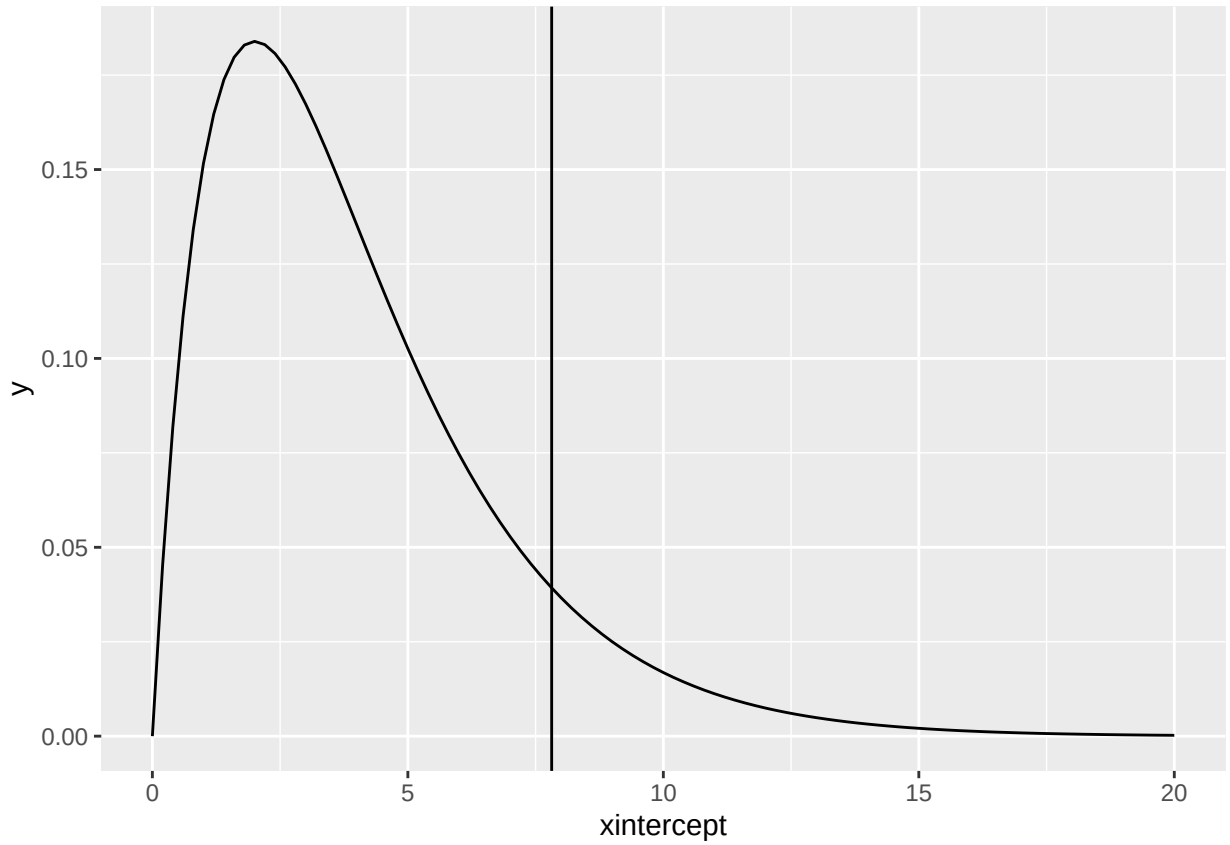
```
ggplot()+
  geom_function(fun = function(x){dchisq(x, 4)})+
  xlim(0, 20)+
  geom_vline(xintercept = qchisq(.95, df = length(p_obs)-1-1))
```




```
# En este caso, el estadístico Q es mayor que el cuantil 0.95 de una chi cuadrada con 4 gdl, por lo que

## Binomial negativa:
Q_nbinom = n*sum(((p_obs-p_nbinom)^2)/p_nbinom)

Q_nbinom
```

```
## [1] 3.125626
```

```
# El valor del estadístico debería de seguir una distribución chi cuadrada con 6 - 2 - 1 = 3 grados de
ggplot()+
  geom_function(fun = function(x){dchisq(x, 4)})+
  xlim(0, 20)+
  geom_vline(xintercept = qchisq(.95, df = length(p_obs)-2-1))
```




```
qchisq(.95, df = length(p_obs)-2-1)
```

```
## [1] 7.814728
```

```
# En este caso, no rechazamos la distribución binomial negativa, con 95% de confianza.

# Podemos también obtener el p-value para el estadístico:
pchisq(Q_nbinom, df = 3, lower.tail = FALSE)
```

```
## [1] 0.3726587
```

```
# En este caso, el p-value es mayor que 0.05, por lo que no podemos rechazar H_0.
```

Aun teniendo una prueba sin rechazar, es conveniente hacer varios grupos o intervalos para nuestra variable y repetir pruebas de bondad de ajuste:

```
# Segunda opción de intervalos:
intervalos = c(0, 300, 380, 400, 450, 500, 550, 700, Inf)
N_grupos = cut(N, intervalos)

## Calculando proporciones empíricas y teóricas:
p_obs = table(N_grupos)/n

p_nbinom = rep(NA, length(p_obs))
```

```r
for(i in 1:(length(p_obs)-1)){
  p_nbinom[i] = pnbinom(intervalos[i+1], r_MLE, p_MLE)-pnbinom(intervalos[i], r_MLE, p_MLE)
}
p_nbinom[length(p_nbinom)] = 1-pnbinom(700, r_MLE, p_MLE)

# Calculamos el estadístico Q_n:

Q_nbinom = n*sum(((p_obs-p_nbinom)^2)/p_nbinom)

# Calculando el p-value para una chi cuadrada con k-d-1 = 8-2-1 = 5 grados de libertad

1-pchisq(Q_nbinom, df = length(p_obs)-2-1)
```

```
## [1] 0.2938975
```

```r
# Tampoco podemos rechazar al 95% que los datos vengan de una distribución binomial negativa.

## Tercera alternativa:

intervalos = c(0, 400, 500, 600, 700, Inf)
N_grupos = cut(N, intervalos)

## Calculando proporciones empíricas y teóricas:
p_obs = table(N_grupos)/n

p_nbinom = rep(NA, length(p_obs))
for(i in 1:(length(p_obs)-1)){
  p_nbinom[i] = pnbinom(intervalos[i+1], r_MLE, p_MLE)-pnbinom(intervalos[i], r_MLE, p_MLE)
}
p_nbinom[length(p_nbinom)] = 1-pnbinom(700, r_MLE, p_MLE)

# Calculamos el estadístico Q_n:

Q_nbinom = n*sum(((p_obs-p_nbinom)^2)/p_nbinom)

# Calculando el p-value para una chi cuadrada con k-d-1 = 5-2-1 = 2 grados de libertad

1-pchisq(Q_nbinom, df = length(p_obs)-2-1)
```

```
## [1] 0.2119315
```

```r
# Tampoco podemos rechazar la distribución
```

Ya con suficientes intentos de distinto número de grupos y de tamaño en los grupos, podemos tener más confianza en la prueba de bondad de ajuste, por lo que nos quedaríamos con la distribución Negativa Binomial con los parámetros estimados como una buena candidata.

Ya teniendo un modelo no rechazado, podemos utilizarlo para hacer cálculos con nuestra variable aleatoria:

Por ejemplo, ¿cuál es la probabilidad de que haya más de 900 choques en el Área Metropolitana de Guadalajara en un mes?

```r
pnbinom(900, size = r_MLE, prob = p_MLE, lower.tail = F)
```

```
## [1] 0.01872428
```

```r
# La probabilidad empírica era la siguiente:
mean(N>900)
```

```
## [1] 0.01388889
```

Podemos también calcular algunos cuantiles:

```r
# Mediana
qnbinom(0.5, r_MLE, p_MLE)
```

```
## [1] 489
```

```r
### empírica:
median(N)
```

```
## [1] 518.5
```

```r
# Cuantil 0.95
qnbinom(0.95, r_MLE, p_MLE)
```

```
## [1] 801
```

```r
### empírico:
quantile(N, 0.95)
```

```
##   95%
## 786.2
```

```r
# 0.99
qnbinom(0.99, r_MLE, p_MLE)
```

```
## [1] 960
```

```r
### empírico:
quantile(N, 0.99)
```

```
##    99%
## 879.24
```

Ejemplo distribución continua:

Generamos una muestra de datos que vienen de una distribución normal con media 5 y varianza 10:

```r
set.seed(8)
X = rnorm(570, mean = 5, sd = sqrt(10))
```

Vamos a estandarizar la distribución muestral para comparar con una normal estándar:

```r
Z = scale(X)
mu = mean(Z)
sigma = sd(Z)
```

Podemos realizar una prueba χ^2:

```r
# Escogiendo mis intervalos:
intervalos = c(-Inf, -2, -1, 0, 1, 2, Inf)
Z_grupos = cut(Z, intervalos)

# Calculando las proporciones empíricas:
n = length(Z)
p_obs = table(Z_grupos)/n

# Proporciones teóricas:
p_norm = c(
  pnorm(-2),
  pnorm(-1)-pnorm(-2),
  pnorm(0)-pnorm(-1)
)
p_norm = c(p_norm, p_norm[3:1])

# Así, podemos obtener nuestro estadístico:

Q_n = n*sum(((p_obs-p_norm)^2)/p_norm)

Q_n
```

```
## [1] 0.8548985
```

```r
# Comparamos con una distribución chi cuadrada con k-d-1 grados de libertad, que en este caso son 6-2-1
qchisq(.95, df = 3)
```

```
## [1] 7.814728
```

```r
1-pchisq(Q_n, df = 3)
```

```
## [1] 0.8362959
```

```r
# No rechazamos la distribución normal estándar para Z.
```

Si quisiera calcular la probabilidad de que X sea mayor que 8:

$P(X > 8) = P(\frac{X-\bar{X}}{S} > \frac{8-\bar{X}}{S}) = P(Z > \frac{8-\bar{X}}{S})$

```r
1-pnorm((8-mean(X))/sd(X))
```

```
## [1] 0.1612875
```

Si hubiera propuesto una distribución distinta, podría ocurrir que rechace la hipótesis nula, que en este caso estaríamos en lo correcto, pero también podríamos no rechazarla, aun cuando los datos vengan de una distribución normal.

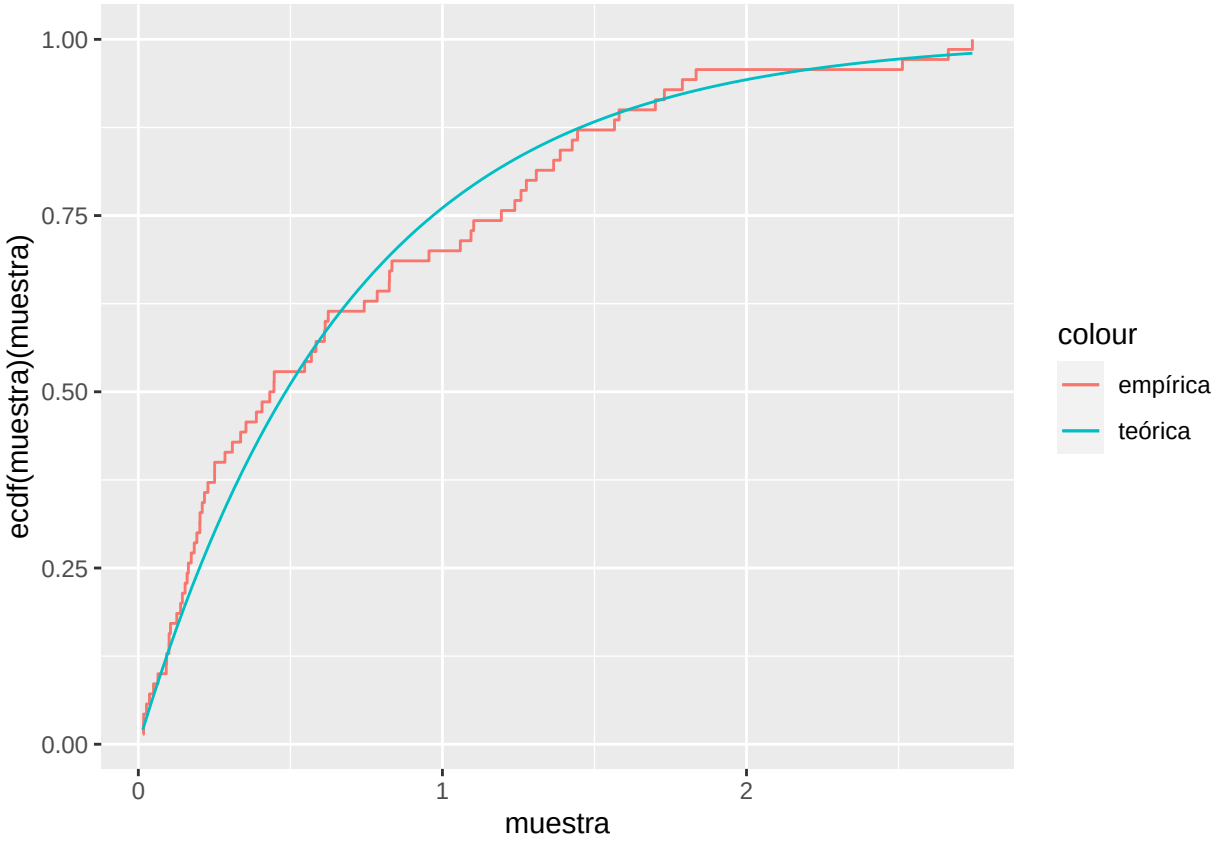
Desventajas de la prueba χ^2:

- Sensible a la elección del número y del tamaño de los grupos.
- No es muy poderosa cuando el tamaño de muestra es pequeño.
- Existen muchas pruebas más poderosas que esta.

Prueba Kolmogorov-Smirnov (KS)

Esta prueba compara dos funciones de distribución. Para nuestra aplicación de bondad de ajuste, compararemos la función de distribución empírica con la teórica.

```r
muestra = rexp(70, rate = 1.4)

ggplot()+
  geom_step(aes(color = "empírica", muestra, ecdf(muestra)(muestra)))+
  geom_function(aes(color = "teórica"), fun = function(x){pexp(x, rate = 1/mean(muestra))})
```



Esta prueba se basa en el siguiente estadístico:

$D_n = sup(|EF_X(x) - F_X(x)|)$

La prueba de hipótesis es la siguiente:

$H_0 : EF_x(X) = F_X(x)$

$H_1 : EF_x(X) \neq F_X(x)$

Si la hipótesis nula es verdadera, entonces:

$\sqrt{n}D_n \to sup(|B(F(t))|)$

Si $K = sup(|B(F(t))|)$, el supremo de un puente browniano, entonces su función de distribución es la siguiente:

$P(K \leq x) = 1 - 2\sum_{k=1}^{\infty}(-1)^{k-1}e^{-2k^2x^2}$

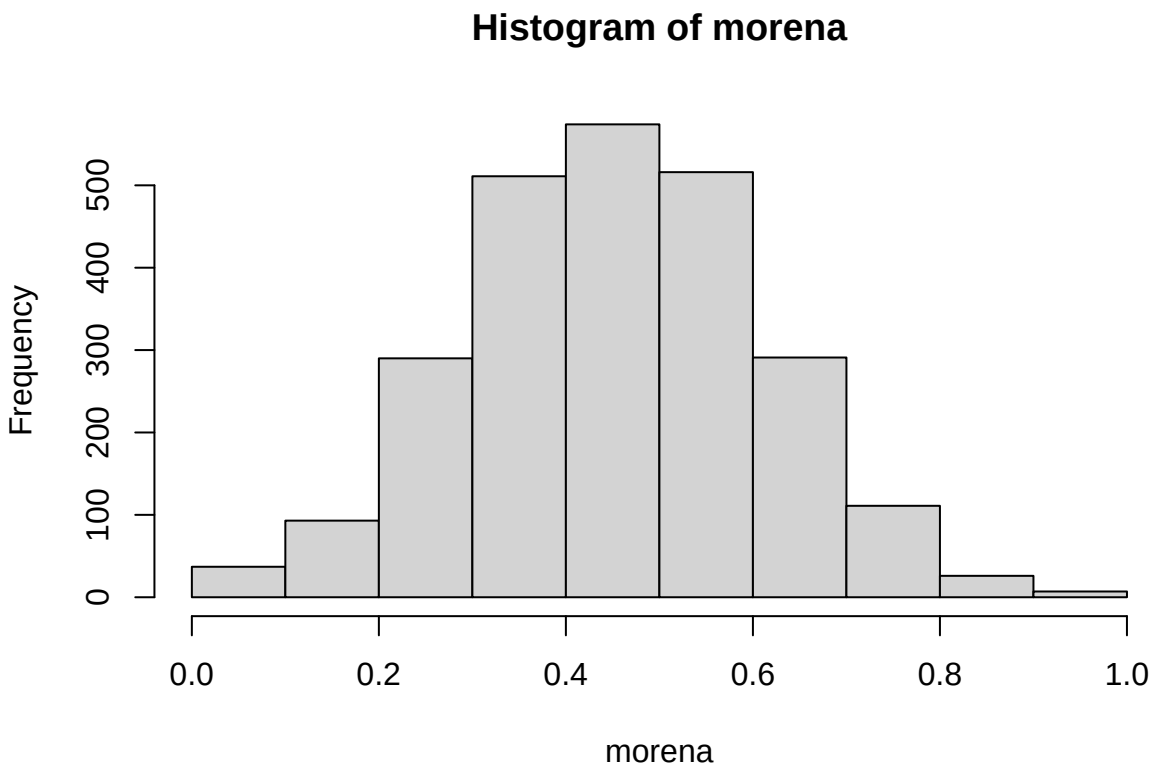
Desde esta función podemos obtener nuestro valor p, que en caso de ser menor que el nivel de rechazo, rechazamos la hipótesis nula y decimos que hay evidencia que nos dice con cierto nivel de confianza que los datos no vienen de la distribución propuesta.

Ejemplo: % de votos por sección electoral para el partido Morena en el estado de Oaxaca

```
bd = read.table("D:/UAG/Modelación actuarial/2022/clases/18. Bondad de ajuste/diputaciones.txt", sep =

oaxaca = bd %>%
  filter(NOMBRE_ESTADO == "OAXACA") %>%
  group_by(SECCION) %>%
  summarise(morena = sum(MORENA), total = sum(TOTAL_VOTOS_CALCULADOS)) %>%
  mutate(porcentaje = morena/total) %>%
  drop_na(porcentaje)

morena = oaxaca$porcentaje
```

Ya teniendo nuestra variable, podemos explorarla y pensar qué distribuciones podríamos usar para modelarla:

```
hist(morena)
```




```
summary(morena)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.005405 0.341613 0.452946 0.452948 0.561539 0.959135
```

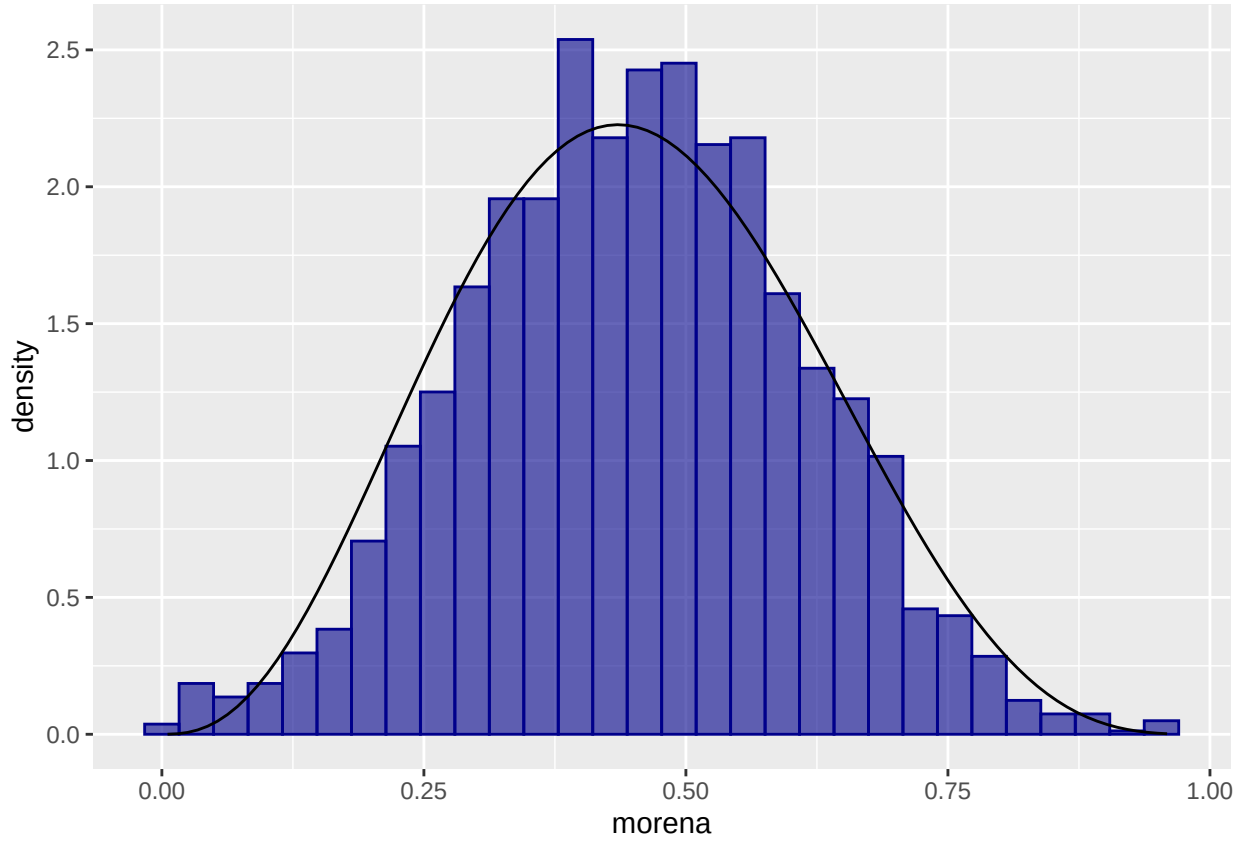
Podríamos usar la distribución beta o la logit-normal:

Estimando los parámetros para estas distribuciones:

```
#Beta:
alpha_MM = (mean(morena)*(1-mean(morena))/var(morena)-1)*mean(morena)
beta_MM = (mean(morena)*(1-mean(morena))/var(morena)-1)*(1-mean(morena))

MLE = optim(par = c(alpha_MM, beta_MM),
            x = morena,
            fn = function(x, par){
              -sum(log(dbeta(x, par[1], par[2])))
            }, method = "L-BFGS-B",
            lower = c(1e-5, 1e-5),
            upper = c(100, 100))
alpha_MLE = MLE$par[1]
beta_MLE = MLE$par[2]
```

Ahora, evaluando gráficamente la distribución:

```
ggplot()+
  geom_histogram(aes(morena, y = ..density..),
                 color = "darkblue",
                 fill = "darkblue",
                 alpha = 0.6,
                 bins = 30)+
  geom_function(fun = function(x){
    dbeta(x, alpha_MLE, beta_MLE)
  })
```



Podemos hacer ahora la prueba KS para bondad de ajuste:

```
ks.test(morena, "pbeta", shape1 = alpha_MLE, shape2 = beta_MLE, alternative = "two.sided")
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  morena
## D = 0.02938, p-value = 0.02882
## alternative hypothesis: two-sided
```

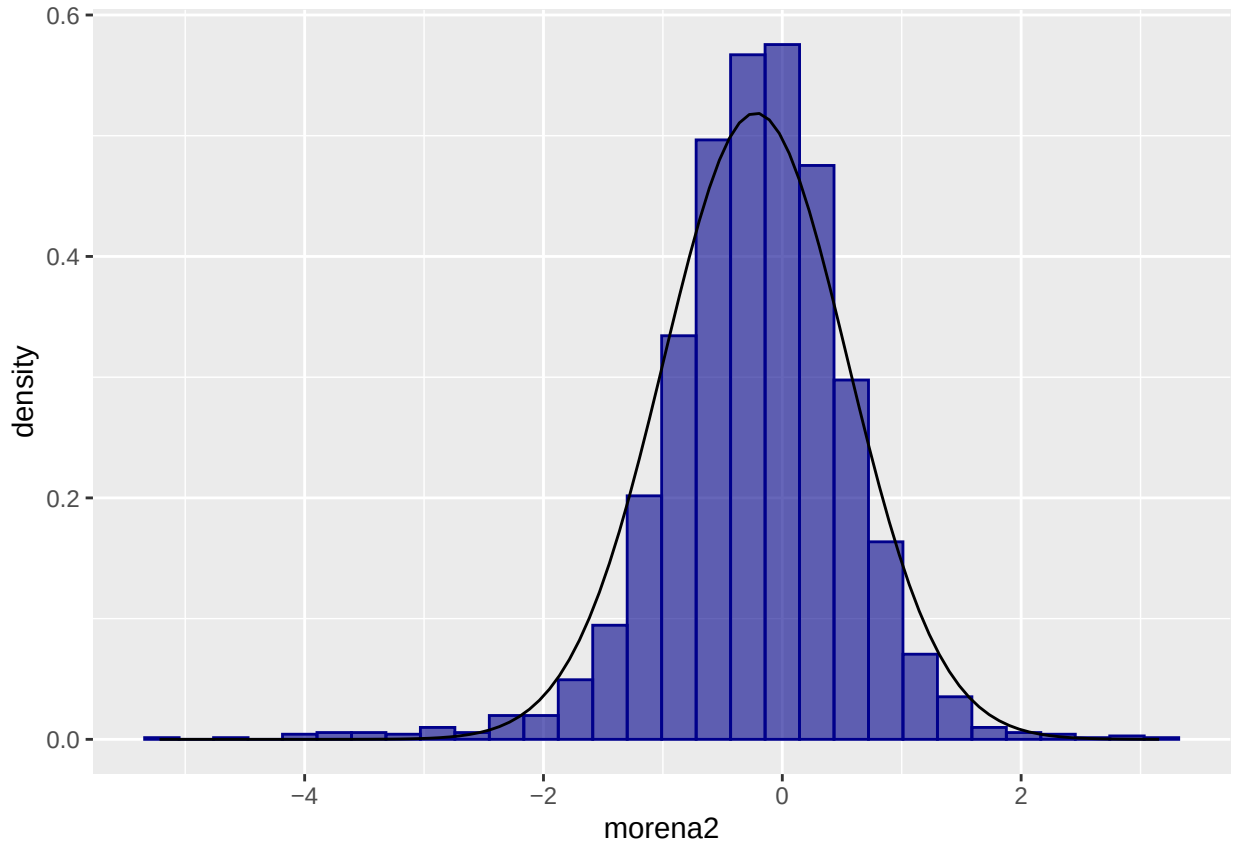
Como nuestro valor p es menor que 0.05, rechazamos la hipótesis nula y según esta prueba, nuestros datos no vienen de una distribución beta con los parámetros estimados.

Estimando los parámetros para la logit-normal:

```r
# Podemos transformar la variable morena para modelar una distribución normal, en lugar de una logit-no

morena2 = log(morena/(1-morena))

# Estimamos los parámetros para una normal por MLE:
mu = mean(morena2)
sigma = sd(morena2)
```

Podemos verificar la bondad del ajuste:

```r
ggplot()+
  geom_histogram(aes(morena2, y = ..density..),
                 color = "darkblue",
                 fill = "darkblue",
                 alpha = 0.6,
                 bins = 30)+
  geom_function(fun = function(x){
    dnorm(x, mu, sigma)
  })
```



Ahora, aplicando la prueba KS:

```r
ks.test(morena2, "pnorm", mean = mu, sd = sigma, alternative = "two.sided")
```

```
##
```

```
##  One-sample Kolmogorov-Smirnov test
##
## data:  morena2
## D = 0.042632, p-value = 0.0002654
## alternative hypothesis: two-sided
```

También rechazamos la distribución logit-normal.

Ventajas de la prueba KS:

- Toma en cuenta toda la función de distribución, considerando la forma y la escala de la variable.
- Es relativamente sencilla de aplicar.
- Más poderosa que la χ^2 con un tamaño de muestra pequeño.

Desventajas de la prueba:

- Requiere de un tamaño de muestra relativamente grande para ser confiable.
- No es tan poderosa como otras pruebas, por ejemplo la Anderson-Darling.
- Es más sensible en la mitad de la distribución y no toma tanto en cuenta las colas.
- No toma en cuenta que los parámetros fueron estimados a partir de los datos.

Prueba Anderson - Darling (AD): Esta es una modificación de la prueba KS, que le da mayor peso a las colas de la distribución y permite tomar en cuenta si los parámetros fueron estimados a raíz de la muestra.

Una desventaja de esta prueba es que es necesario calcular unaa distribución asintótica distinta para cada distribución paramétrica propuesta. Sin embargo, también es mucho más poderosa con tamaños de muestra pequeños.

Sólo se utiliza para variables continuas.

Se basa en el siguiente estadístico:

$A^2 = -n - S$

Donde

$S = \sum_{i=1}^{n} \frac{2i-1}{n}[ln(F(Y_i) + ln(1 - F(Y_{n+1-i})]$

Considerando a Y_i como la muestra ordenada de manera ascendente.

Si la hipótesis nula es verdadera, el estadístico seguirá una distribución teórica particular. Ésta dependerá de la distribución paramétrica propuesta.

Ejemplo:

Para las distribuciones modeladas anteriormente, podemos realizar esta prueba:

```
library(goftest)

# Beta:
ad.test(morena, null = "pbeta", shape1 = alpha_MLE, shape2 = beta_MLE, estimated = TRUE)
```

```
##
##  Anderson-Darling test of goodness-of-fit
##  Braun's adjustment using 50 groups
##  Null hypothesis: beta distribution
##  with parameters shape1 = 3.67838486193229, shape2 = 4.47697000138493
##  Parameters assumed to have been estimated from data
##
## data:  morena
## Anmax = 4.3051, p-value = 0.2695
```

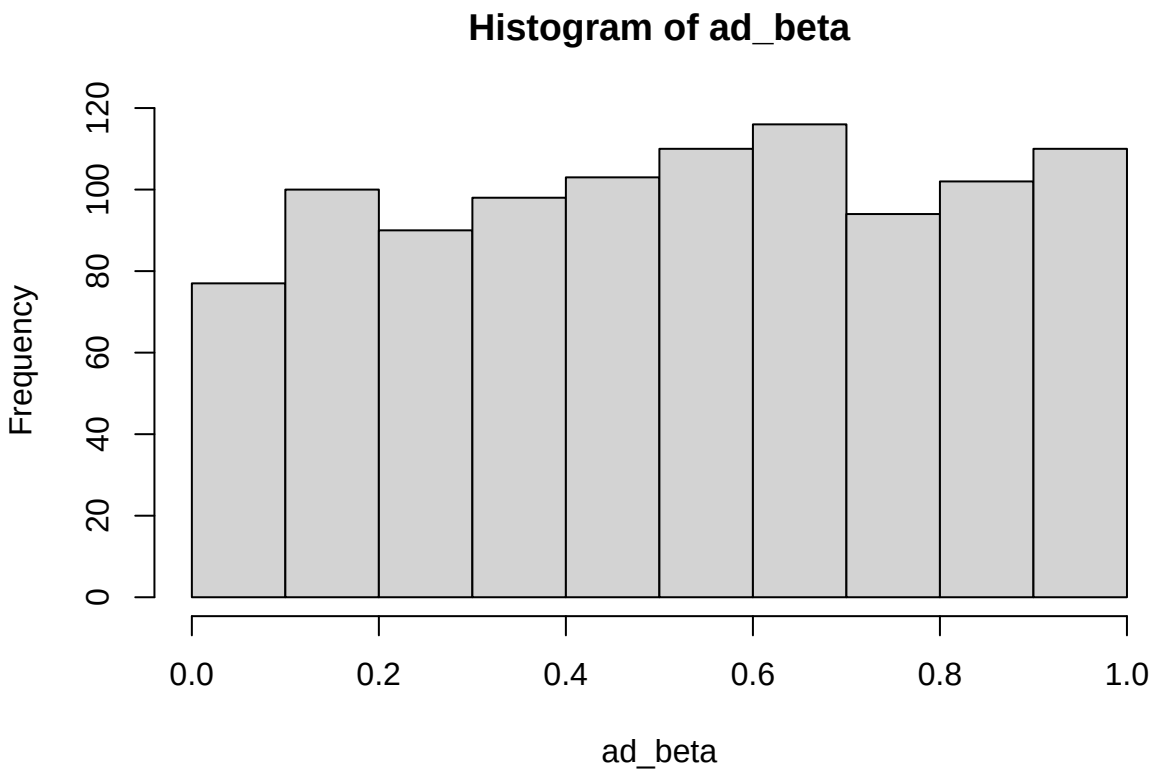
```
# Me da un p-value mayor que .05, por lo que no podemos rechazar la distribución beta.

# Logit-normal:
ad.test(morena2, null = "pnorm", mean = mu, sd = sigma, estimated = TRUE)
```

```
##
##  Anderson-Darling test of goodness-of-fit
##  Braun's adjustment using 50 groups
##  Null hypothesis: Normal distribution
##  with parameters mean = -0.222696763009665, sd = 0.768948414201536
##  Parameters assumed to have been estimated from data
##
## data:  morena2
## Anmax = 2.514, p-value = 0.9187
```

```
# tampoco podemos rechazar la distribución logit-normal
```

Por considerar que los parámetros fueron estimados mediante la muestra, el ajuste de Braun provocará que el valor p sea distinto en cada prueba, por la aleatoriedad del ajuste. Por esto, es buena práctica realizar muchas pruebas y ver cómo se distribuye el valor p.

```
# Beta:
ad_beta = replicate(
  n = 1000,
  ad.test(morena, null = "pbeta", shape1 = alpha_MLE, shape2 = beta_MLE, estimated = TRUE)$p.value
)

# Podemos explorar la distribución de los valores p:
hist(ad_beta)
```



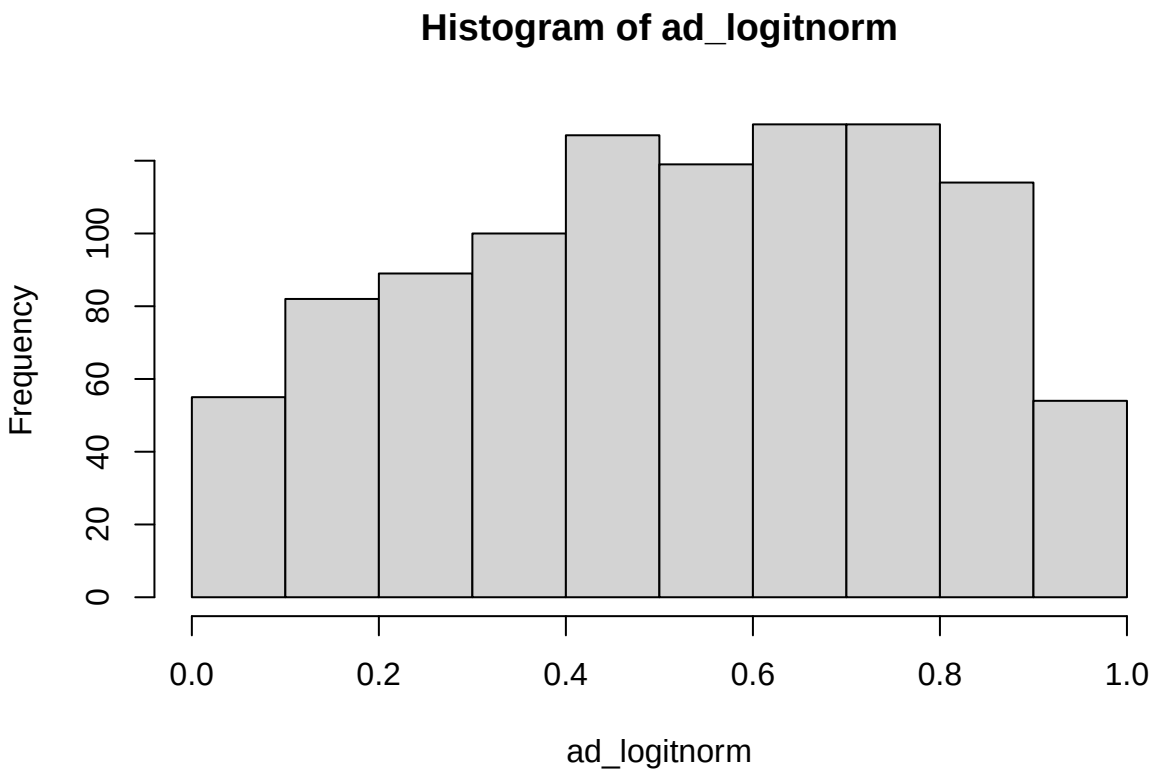

```
summary(ad_beta)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## 0.0006651 0.2827494 0.5239255 0.5194651 0.7594733 0.9968276
```

```
mean(ad_beta>.05)
```

```
## [1] 0.959
```

```
# Logit-normal:

ad_logitnorm = replicate(1000,
                         ad.test(morena2, null = "pnorm", mean = mu, sd = sigma, estimated = TRUE)$p.val
)

# Podemos explorar la distribución de los valores p:
hist(ad_logitnorm)
```




```
summary(ad_logitnorm)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002739 0.320490 0.547060 0.525170 0.734770 0.991216
```

```
mean(ad_logitnorm>.05)
```

```
## [1] 0.981
```

Con estos resultados, no podríamos rechazar que el porcentaje de votos por sección por el partido Morena en Oaxaca vengan de una distribución Beta o logit-normal.

Ahora, ¿cómo podemos comparar entre estos modelos para saber cuál es mejor?